



FOUR Js
The Power of Simplicity

Four Js Genero Studio User Guide



Contents

What's new in Genero Studio, v 3.00.....	11
Welcome to Genero Enterprise.....	15
What is Genero Enterprise.....	15
Tour Genero Studio.....	20
Running the Demos.....	32
Switching Genero Clients.....	36
Creating with Quick Starts.....	38
Finding more information.....	39
Welcome to Genero Studio.....	40
Getting Started with Genero Studio.....	41
Quick Start: Tour of Genero Studio.....	41
Run the OfficeStore demo.....	41
Explore the Debugger.....	44
vi emulator and diff tools in Code Editor.....	51
Explore Source Code Management.....	52
Explore database meta-schemas.....	59
Explore forms.....	60
Analyze code.....	65
Generate code.....	69
The Genero Studio framework.....	69
The Welcome Page.....	70
Modules.....	71
Toolbars and Menus.....	82
Views.....	96
Dialogs.....	100
Learning to use Genero Studio.....	103
Command line options.....	103
Samples directory.....	103
Integrating existing applications.....	105
Setting Preferences.....	106
Access Help.....	113
Creating with Quick Starts.....	114
Configuring Genero Studio.....	115
Software configuration scenarios.....	115
Default configuration.....	117
Change the active configuration.....	117
Configuring for BAM.....	118
Configure for Genero Mobile.....	118
Configure Genero Mobile for Android.....	118

Configure Genero Mobile for iOS.....	127
Display to the Genero Mobile Development Client.....	138
Setting up a local environment.....	139
Local environment software requirements.....	140
Define local Genero installations, GAS configurations, and environment sets.....	140
Environment sets.....	140
GST-specific environment variables.....	143
Add or edit environment variables.....	144
Display clients.....	146
Desktop: GDC configurations.....	146
Web: GAS/GWC configurations.....	147
Mobile clients: GMI and GMA configurations.....	153
Setting up a remote environment.....	154
Remote environment software requirements.....	154
Add a remote host.....	155
Define mount points to shared drives.....	161
Define remote Genero installations, GAS configurations, and environment sets.....	162
Share projects / source code management.....	162
Access a database.....	163
Language support (text encoding).....	163
Configure Genero Studio to use a different character set.....	164
Add a text encoding plugin.....	165
Character mapping table (encodingMap.xml).....	167
Configure keyboard and language on a Windows™ client.....	168
Configure LANG on a Genero Studio Server.....	168
Test text encoding configuration.....	170
Configuration reference.....	170
Genero Configuration Management dialog.....	171
Compiler / Runtime configuration (Genero Installations).....	172
Import Configuration dialog.....	173
Import Preferences dialog.....	173
GAS standalone dispatcher: httpdispatch.....	174

Business Application Modeling (BAM).....176

Quick Start: Generate an application.....	176
Create a managed project.....	176
Add a meta-schema to the project (4dbx).....	177
Create the Business Application diagram (4ba).....	177
Implement the program and form.....	178
Generate and run the application.....	179
Add a detail list to the form.....	180
Quick Start: Generating a mobile app.....	182
Create a BAM mobile project.....	182
Create a database.....	184
Create form from database.....	187
Generate and run the app.....	187
Add phone functionality to the app.....	188
Customize the app.....	190
Package and Deploy.....	191
BAM Concepts.....	192
What is Business Application Modeling (BAM)?.....	192
How code is generated.....	194
The modeling diagrams.....	196
Mobile apps vs Desktop applications.....	198
The default template features.....	199

Configuring for BAM.....	200
BAM Projects.....	200
Managed projects.....	201
Mobile projects.....	201
Modeling the application.....	202
The Business Application (BA) diagram.....	202
Create a BA diagram.....	204
Add and implement a program.....	204
Add Forms.....	205
Add Reports.....	210
Add Web services (Server, Services, Forms with services).....	215
Add Relations.....	220
Add mobile device features (Photo, Gallery, Phone, Mail, SMS, Contact, Maps, Barcode)..	223
Import files into the diagram from the project.....	225
Modeling the database.....	226
Database meta-schema (4dbx).....	226
Create a meta-schema.....	227
Extract meta-schema information from database.....	228
Add a meta-schema to a project.....	231
Managing SERIALs in a generated application.....	231
Managing concurrency.....	231
Cascade delete.....	232
Working with forms.....	232
Mobile forms.....	233
Enable and disable CRUD logic.....	239
Form behavior in CRUD states.....	241
Data refresh.....	243
Control the row position in form.....	244
Opening a form with a subset of data.....	244
Field activation.....	244
Define queries and data order.....	245
Define a dynamically populated ComboBox.....	245
Lookup fields.....	247
Add buttons to form.....	249
Add formonly (nondatabase) fields to a form.....	250
Master-detail forms.....	250
Adding custom code.....	250
Understanding what gets generated.....	250
Finding the right place to customize.....	259
Using POINTs and BLOCKs.....	262
Modifying the look and feel.....	264
Default actions.....	264
Modify action defaults (dbapp.4ad).....	266
Default Topmenu and Toolbar.....	267
Modify the Topmenu (dbapp.4tm).....	267
Modify the Toolbar (dbapp.4tb).....	267
Modify styles (dbapp.4st).....	267
BAM Reference.....	268
BAM-specific environment variables.....	268
\$(generate).....	268
tclsh.....	269
\$(tcl) - deprecated.....	270
\$(blockpoint).....	270
Business Application Modeling error messages.....	271
Business Records error messages.....	280
Business Application Diagram error messages.....	285

Meta-schema Manager.....	288
What is a database meta-schema? (4db).....	288
Creating a meta-schema.....	289
Create a meta-schema.....	289
Extract meta-schema information from database.....	289
BDL schema file (sch).....	293
Add a meta-schema to a project.....	293
Adding more information to a meta-schema.....	293
Add new tables and columns.....	294
Add constraints or indexes.....	294
Add foreign keys.....	295
Add a many-to-many relationship.....	298
Manage SERIALs.....	300
Centralize field information (label, widget, default value).....	300
Viewing a meta-schema.....	300
Comparing two meta-schemas.....	302
Update a meta-schema from database.....	302
Generate a database script from meta-schema.....	303
Generate meta-schema documentation.....	305
Meta-schema Manager Reference.....	305
Meta-schema properties.....	305
Data types.....	306
Database server/user information.....	308
Dialogs.....	309
Views.....	315
Meta-Schema Manager preferences.....	317
Meta-schema diagram context menu.....	318
Meta-schema Manager error messages.....	318
DB Explorer.....	332
Open DB Explorer.....	333
Using the Views menu.....	333
From the meta-schema or Data Model.....	333
Change connection details.....	333
Show data.....	334
Show table data (start with meta-schema diagram).....	335
Show table data (start with DB Explorer).....	335
Show data for select columns.....	335
Show data for a business record.....	335
Change the data.....	336
Show data from a table in edit mode.....	336
Show data from a different table in edit mode.....	336
Update data.....	336
Insert a row.....	336
Duplicate a row.....	337
Delete a row.....	337
Limit rows.....	337
Write a SQL query by hand.....	338
Execute a query.....	339
Project Manager.....	340
Genero project file (4pw).....	340

Quick Start: Create a project.....	340
Creating new projects.....	341
Create a new project.....	341
Import existing files as a new project.....	341
mkproject - Convert a Makefile to a project.....	342
Connect to existing build systems.....	342
Organizing projects.....	342
Groups, Applications, and Libraries.....	342
Using external libraries.....	343
Setting external dependencies.....	343
Building and linking programs.....	343
Languages.....	344
What are build rules.....	344
Add/Edit a build rule.....	345
Example: How build rules work.....	346
Link rules.....	347
Execution rules.....	347
Command line options for build, link, execution rules.....	347
Environment variables.....	348
Pre/Post compile.....	351
Pre/Post link.....	351
gsmake - Command line option to build projects.....	352
Packaging.....	353
Locate a file (starting at Project Manager).....	353
Project Manager Reference.....	354
Project Manager context menu.....	354
Dialogs.....	355
Project Manager node properties.....	364
Predefined node variables.....	366
Project Manager error messages.....	369
Code Editor.....	374
Editing code files.....	374
Code Editor basics.....	374
Smart editing - indenting, tabs, and backspace.....	375
Fold text.....	375
Bookmarks.....	376
Auto completion (Ctrl+Space).....	376
Code templates (Ctrl+T).....	376
Split a document.....	377
Square selection.....	377
XML editing.....	377
Search and replace.....	377
Using the Diff tool.....	380
Printing files.....	382
Using XML catalog files.....	383
XML catalog files.....	383
The XML catalog file.....	383
Manage XML catalog entries.....	384
Code Editor Reference.....	384
Code Editor preferences.....	384
Customize Diff tool: preferences.....	390
Views.....	390
Keyboard Shortcuts.....	394
vi Commands List.....	398

Code Analyzer.....	402
Sequence Diagrams.....	402
Dependency Diagrams.....	403
Dependency diagram context menu.....	405
Form Designer.....	406
Forms in Genero applications.....	406
Quick Start: Creating a first form.....	407
Creating the user interface.....	407
Forms.....	408
Business records (data sets).....	411
Containers.....	413
Widgets.....	420
Web components.....	432
Action management (Toolbars, Topmenus).....	434
Styles.....	438
Form Designer usage.....	440
Drawing.....	440
Selecting, moving, resizing.....	440
Aligning.....	441
Transforming.....	441
Command-line syntax: gsform.....	442
Localizing your form.....	443
Form Designer Reference.....	443
Customize Form Designer: preferences.....	444
Form tab.....	445
Business Record diagram.....	446
Menus.....	446
Views.....	447
Dialogs.....	449
Properties list.....	459
Form Designer error messages.....	484
Business Records error messages.....	493
XML validation error messages.....	497
File Browser.....	500
Navigating files in File Browser.....	500
Selecting files in File Browser.....	500
Managing files in File Browser.....	500
Locate a file (starting at File Browser).....	501
Graphical Debugger.....	502
Controlling program execution.....	502
Start the Debugger.....	502
Start the Debugger on a running program.....	502
Debug a Web services server application.....	504
Debug a mobile application.....	504
Stop the Debugger.....	505
Step through the program.....	506
Breakpoints.....	506
Watchpoints.....	507

Debugger (fgldb) command prompt.....	508
Debugger output.....	509
Examining data.....	509
Examining execution flow.....	509
Record/replay a macro.....	510
Profiler.....	510
Local vs. remote debug.....	511
Reference.....	511
Debug context menu.....	511
Views.....	512
Supported debug commands.....	516
Source Code Management - SVN.....	529
What is Genero Source Code Management?.....	529
SCM Usage.....	530
Checkout files.....	530
Add files.....	531
Commit / Review changes.....	531
Locking.....	532
Revert changes / Un-add files.....	534
Delete files.....	534
Update / Update All.....	534
Cleanup.....	535
Copy working files and directories.....	535
Revert from a single revision.....	535
Merge and revert.....	535
Move a working copy (Switch).....	536
Create patch.....	536
Apply patch.....	536
Browse repository.....	537
View log information.....	537
Specify the revision range for logs.....	537
Blame.....	537
Diff with revised file.....	539
Integrate bug tracking.....	539
SCM Reference.....	540
Views.....	540
Dialogs.....	544
Specify a Subversion client.....	549
SVN error messages.....	549
Report Writer.....	551
Get Started with Reports.....	551
Introduction to Reports.....	551
Steps to a Report.....	553
The Reports demo.....	555
Configure fonts and printers.....	559
Create a report program.....	561
Genero BDL and the Report Writer.....	562
Create the data schema.....	655
Generate a data schema from a Genero BDL report program.....	656
Create a report design document.....	656
What's new in Genero Report Designer, v 3.00.....	656
The Report Design Document.....	657

Report Designer Reference.....	722
RTL Class Reference.....	806
Upgrading Genero Report Designer.....	837
Report templates.....	840
Create a report from an existing template.....	840
Create a new report template.....	847
Create a report template from an existing report.....	853
Modify an existing report template.....	854
Set the report template directory.....	854
GenerateReport command options.....	855
Report Writer Deployment and Customization.....	858
GRE environment variables.....	858
Genero Report Viewer for HTML5 customization.....	859
Distributed Mode.....	859
Genero Report Engine error messages.....	861

Web Services..... 910

Create a Web Services program.....	910
Add Web Service.....	910
The Web Services wizard.....	911
Build the application.....	913
Update the WSDL.....	914
Add Web services (Server, Services, Forms with services).....	914

Mobile applications..... 915

Mobile development environment.....	915
Genero mobile app demos.....	917
Debugging a mobile app.....	918
Debug version of a deployed app.....	919
Debug tools for apps in developer mode.....	920
Viewing the AUI tree.....	921
Viewing the program logs.....	923
Viewing the device logs.....	924
Debugging a Web Component.....	925
Localize a mobile app.....	926
Genero Mobile error messages.....	927

BAM Template Developer Guide..... 930

Quick Start: Customizing templates.....	930
Example 1: Adding a new property.....	931
Example 2: Adding a File >> New item.....	934
Example 3: Adding an entity to the BA Diagram.....	936
How code is generated.....	940
The code generation template set.....	942
Interpreting settings.agconf.....	943
Interpreting creatables.conf.....	947
Tcl basics and samples.....	948
Add POINT and BLOCK sections to template.....	949
Example: Using XSLT instead of Tcl.....	949
Image directory structure.....	952
Template Reference.....	953
XML reference.....	953
POINT and BLOCK reference.....	991

Packaging, deploying, and distributing apps.....	993
Genero Archive (GAR) packaging.....	993
Packaging for a mobile device.....	993
What is packaging?.....	994
Packaging process overview for a mobile device.....	996
Package a mobile app.....	996
Deploy a mobile app for testing.....	998
Package and Directory nodes and properties.....	999
Platform: Package and deploy rules.....	1003
Distribute your app.....	1005
Manage App updates.....	1008
Upgrading.....	1010
New Features of Genero Studio.....	1010
What's new in Genero Studio, v 3.00.....	1010
What's new in Genero Studio, v 2.51.....	1014
What's new in Genero Studio, v 2.50.....	1014
What's new in Genero Studio, v 2.41.....	1021
What's new in Genero Studio, v 2.40.....	1021
Upgrade Guides.....	1022
3.00 upgrade guide.....	1023
2.50 upgrade guide.....	1023
2.41 upgrade guide.....	1025
2.40 upgrade guide.....	1026
2.30 upgrade guide.....	1026
2.20 upgrade guide.....	1026
Migrating to a new BAM template set.....	1027
Migrate from dbapp3.2 to dbapp4.0.....	1027
Migrate from dbapp3.1 to dbapp3.2.....	1027
Migrate from dbapp3.0 to dbapp3.1.....	1028
Migrate from dbapp2.0 to dbapp3.0.....	1028
Migrate from dbapp1.0 to dbapp2.0.....	1028
Migrate from 2.3x to dbapp1.0.....	1029
Migrate customized template sets.....	1029
Legal Notices.....	1031

What's new in Genero Studio, v 3.00

This publication includes information about new features and changes in existing functionality.

Important: Please read [What's new in Genero Studio, v 2.51](#) on page 1014, for a list of features that were introduced with the Genero Mobile 1.0 release.

These changes and enhancements are relevant to this publication.

Table 1: General, Version 3.00

Overview	Reference
Genero Studio now supports Genero, Genero Report Writer for BDL, and Genero Mobile from a single installation.	N/A
Genero Studio now supports the Genero Web Client for Javascript (GWC-JS).	See Web: GAS/GWC configurations on page 147.
Genero Studio now supports connecting to the application server using HTTPS.	See Web: GAS/GWC configurations on page 147.
Generate a Genero archive (GAR) from Studio, for deployment to the GAS.	See Genero Archive (GAR) packaging on page 993
Genero Studio gives the ability to locate a document in the File Browser, in a BA diagram, or in the System File Browser (the file browser of the operating system). The new System File Browser feature facilitates the use of system file explorer integrated tools, such as SVN or Git integration.	See Locate a file (starting at Project Manager) on page 353 or Locate a file (starting at File Browser) on page 501.
New configuration option for GAS, allowing the ability to run and debug Web services from Genero Studio.	See Web: GAS/GWC configurations on page 147.

Table 2: Project Manager, Version 3.00

Overview	Reference
The properties Web Service and Web Service URL suffix have been added for the Application node, allowing the ability to run and debug Web services from Genero Studio.	See Project Manager node properties on page 364.

Table 3: DB Explorer, Version 3.00

Overview	Reference
DB Explorer module introduced to view and modify data in database tables and to test SQL query results. With this tool, you can right-click on forms, reports and Web services to view the data.	See DB Explorer on page 332.
DB Explorer expands support of SQL commands, in addition to SELECT, INSERT, UPDATE, and DELETE.	See DB Explorer on page 332 and Write a SQL query by hand on page 338.

Table 4: Meta-Schema Manager, Version 3.00

Overview	Reference
Enhanced schema view displays database schema modifications at a glance.	See Viewing a meta-schema on page 300.
Mouse over items in the schema for more detail, to include a summary of schema changes, primary key column details, and more.	See Viewing and manipulating a meta-schema .
Reorder columns using drag-and-drop.	See Viewing and manipulating a meta-schema .
Move columns to another table using drag-and-drop.	See Viewing and manipulating a meta-schema .
HTML meta-schema documentation provides details of all database objects and facilitates global schema review.	See Generate meta-schema documentation on page 305.
Toggle label display shows or hides foreign key names in the diagram.	See Viewing and manipulating a meta-schema .

Table 5: Genero Mobile, Version 3.00

Overview	Reference
You can debug an application deployed to a mobile device. With this new feature, the application is running on the mobile device and the Graphical Debugger is able to attach to the process.	See Debug a mobile application on page 504
The <i>DBAPP_MOBILE</i> environment variable provides warning messages regarding features not supported by mobile devices during the compilation of applications generated by the Business Application Modeler.	See DBAPP_MOBILE on page 268.

Table 6: Business Application Modeler, Version 3.00

Overview	Reference
Publish JSON Web services via the Business Application Modeler.	See JSON Web services on page 220.
SOAP Web services enhanced with XML and XSD Schema Serialization attributes.	See Webservice entity on page 217.

Table 7: Code Editor, Version 3.00

Overview	Reference
Code Editor supports a horizontal view in the Diff tool, in addition to the vertical view of previous versions.	See Using the Diff tool on page 380.

Table 8: Form Designer, Version 3.00

Overview	Reference
Support for new Form properties: <code>keyboardHint</code> , <code>completer</code> , <code>wantFixedPageSize</code> , <code>action</code> , <code>Disclosure Indicator</code>	See Properties list on page 459.
Support for new <code>DateTimeEdit</code> widget.	See DateTimeEdit on page 425.

Table 9: Search, Version 3.00

Overview	Reference
Search Results pane displays an improved search view, to include previous and current search results organized as a collapsible tree.	See The Search Results view on page 393.

Table 10: Genero Report Writer, Version 3.00

Overview	Reference
Genero Report Viewer for HTML5 provides a browser equivalent of the Genero Report Viewer.	See fgl_report_selectDevice on page 626.
A command-line utility checks and upgrades report design documents (.4rpt) files in batch.	See Upgrading reports from prior versions on page 837.
Report templates provide a wizard-based method for creating report design documents (.4rpt) from a generic report design. The wizard allows you to bind repeating sections, add fields, and bind placeholders and parameters from a data schema, in order to create a stand-alone report design document. A library of report templates have been provided, and you can create your own templates. A template expansion mechanism is available as a command line tool, usable from applications for generic reports.	See Report templates on page 840.
The report engine now limits the number worker threads in distributed mode, to prevent memory exhaustion in times of critical load. Change the default value (25 threads) with the environment variable GRE_MAX_CONCURRENT_JOBS.	See GRE_MAX_CONCURRENT_JOBS on page 858.
You can now configure the default output directory for the Genero Report Engine with the GREOUTPUTDIR environment variable.	See GREOUTPUTDIR on page 858.
There is an improved architecture using HTTP for previewing documents in a distributed setup. Besides improvements in performance, the solution no longer requires the installation of a DVM on the remote machine.	See Distributed Mode on page 859.

Table 11: Genero BDL Reporting APIs

Overview	Reference
APIs support the Genero Report Viewer for HTML5:	See fgl_report_setBrowserDocumentDirectory on page 629, fgl_report_setBrowserDocumentDirectoryURL on page 629, fgl_report_setBrowserFontDirectory on page 630, fgl_report_setBrowserFontDirectoryURL on page 630
A new API supports distributed mode.	See fgl_report_configureDistributedURLPrefix on page 614.
APIs have been introduced to get error details.	See Functions to get error details on page 653.

Overview	
A new API can programmatically set the value of environment or user-defined variables.	See fgl_report_setEnvironment on page 627.

Table 12: Genero Report Designer, Version 3.00

Overview	Reference
Genero Report Designer provides a LastPageFooter section property.	See section (Section) on page 754.
Support of Intelligent Mail bar code type.	See intelligent-mail on page 795.
New <code>smartParse</code> bar code property for bar code Code-128. When enabled, this allows you to enter the bar code value, and the internal code will be computed for you resulting in the shortest visual representation.	See smartParse (Smart Parse) on page 755 and code-128 on page 769.
New <code>gs1*</code> bar code aliases.	See Bar Code type listing on page 765.

Table 13: Graphical Debugger, Version 3.00

Overview	Reference
You can debug an already running process by attaching to the process. The process can be running locally or on a remote computer. Attaching to a remote process allows you to debug an application at a production site where Genero Studio is not installed.	See Start the Debugger on a running program on page 502.
You can debug an application deployed to a mobile device. With this new feature, the application is running on the mobile device and the Graphical Debugger is able to attach to the process.	See Debug a mobile application on page 504
You can debug Web services: server, client or both.	See Debug a Web services server application on page 504.

Welcome to Genero Enterprise

Welcome to Genero, the software infrastructure for enterprise business application development and deployment.

When you first install Genero Enterprise and open Genero Studio, you see the Welcome page, which provides access to the projects, tutorials and samples. You can start by running the demos or by creating applications using the Quick Starts.

What is Genero Enterprise

Genero Enterprise provides an integrated development environment that you can use for the creation of applications to deploy to desktop, web, cloud, and mobile clients.

Genero Enterprise Components

Genero Enterprise is made up of visual tools, business and program logic components that effectively work together to quickly create applications and reports.

Genero Business Development Language (BDL)	Genero BDL is a simple, easy-to-learn programming language for data-intensive business applications. See Genero Business Development Language .
Genero Studio	Genero Studio is an intuitive suite of visual tools (or modules) for creating application interfaces and developing and debugging the underlying program logic. See Genero Studio .
Genero Report Writer	Genero Report Writer is a set of programming APIs and a graphical user interface (GUI) that provide a drag-and-drop interface for the design of business reports, such as corporate or accounting documents, pre-printed forms, labels, and business graphs. See Genero Report Writer .
XML-based Abstract Presentation Layer	Genero provides an abstract definition of the user interface as an XML tree of objects that can be manipulated at runtime by the front end client to enable GUI independence for your Genero application. See XML-based Abstract Presentation Layer .
Dynamic Virtual Machine (DVM)	The DVM executes your application and manages database interaction and communications with client platforms. See DVM .
Open Database Interface (ODI)	The ODI refers to the component of the Genero DVM that maps database-vendor-independent high-level code to provide native access for a variety of vendor databases. See ODI .
Genero Web Services (GWS)	Web services enables applications built with different technologies to integrate and exchange data across varied systems and enterprises using web-based protocols. See GWS .

Genero Desktop Client (GDC)

The GDC provides a graphical front end that displays your Genero application on Windows™, Linux™, or Apple™ desktops. See [GDC](#)

Genero Application Server (GAS)

The GAS provides remote access to Genero applications deployed for the GDC. The GAS includes a Web client component - the Genero Web Client (GWC) - to support the development of database-oriented web applications. The GAS also manages Genero Web Services. See [GAS](#).

Note: This installation of the GAS does not include the web server component of the software, however the products are fully functional standalone and can be used for developing applications. For production systems, you can install the web server component later using the installation packages of the full GAS products.

These components are discussed in more detail below.

Genero Business Development Language (BDL)

Genero BDL is a simple, easy-to-learn programming language for data-intensive business applications. Use Genero BDL to build an interactive database application, a program that handles the interaction between a user and a database.

The programming language is English-like and easy-to-learn. Executable statements enable:

- Program flow control
- Conditional logic
- Error handling
- Structured Query Language (SQL) statement support

High-level program instructions substitute for the many lines of code usually needed to handle user interaction and database manipulation. For example, the `INPUT BY NAME` instruction turns program control over to the user, and allows the user to move around the application form, entering or modifying data.

SQL statements to communicate with database servers are a part of BDL. Dynamic SQL management allows you to execute any SQL statement that is valid for your database version, in addition to those included as part of the BDL.

Built-in classes and predefined functions support BDL features and enhance rapid application development. Using the Genero classes and functions, you can manipulate your application's user interface at runtime and perform other common application tasks.

Data types supported by the language include user types, which you can be defined as synonyms of existing data types, or as shortcuts for records and array structures.

XML-based Abstract Presentation Layer

A Genero application presents its user interface as windows and forms. Interactive elements on the forms (such as buttons, toolbars, and menus) allow the user to trigger actions within the application.

Forms can be designed using the graphical Genero Studio [Form Designer](#), directly in text files, or built dynamically by the application.

The XML-based Abstract Presentation Layer allows a single Genero source code stream to support different user interfaces. **Genero client** software, such as the Genero Desktop Client or the Genero Web Client, display an application's interface on the client machines.

Figure 1: How the AUI tree functions on page 17 shows how the abstract user interface (AUI) tree is shared by the runtime system and the front end client. Their interaction during the running of the application is explained here.

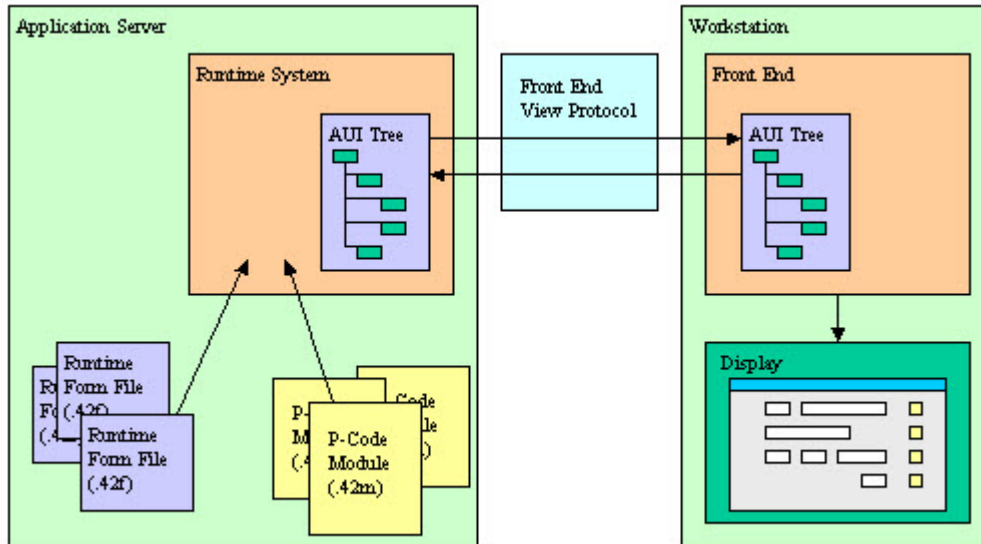


Figure 1: How the AUI tree functions

- The form definition files are translated into XML documents when they are compiled into **runtime form files**.
- The **Genero runtime system** creates the **AUI tree** from the XML documents, and sends this information to the **Genero client** software.
- The Genero client software uses the AUI tree to display the application's interface on the client machine.
- When a user triggers an action, the event is transmitted to the runtime system for interpretation.
- Any changes to the user interface resulting from this action are applied to the AUI tree.
- The runtime system automatically synchronizes the copies of the AUI trees on the application server and the client machine.

Important: Genero BDL contains built-in classes that allow an application to modify the application's interface at runtime, dynamically changing the appearance of the application.

Dynamic Virtual Machine

The Dynamic Virtual Machine is the software or runtime system (`fglrun`) where an application's business logic is processed. It serves as a highly efficient application server that:

- Manages communications with clients.
- Supports a wide variety of architectures, including the Web and Web services.
- Optimizes performance across multiple platforms and databases.

The DVM's n-tier architecture enables the distribution of applications and databases through firewalls and across distributed networks.

Ported to multiple UNIX, Windows, and Linux platforms, the Dynamic Virtual Machine executes the portable byte code (P-code) of Genero applications.

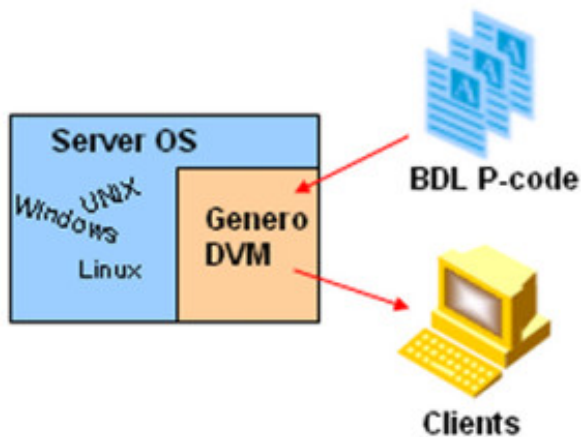


Figure 2: Dynamic Virtual Machine executes p-code

Open Database Interface

Using the Genero Open Database Interface architecture a Genero application can connect to database servers from different database vendors, including Oracle™, IBM™ DB2™, PostgreSQL™, SQL Server™, MySQL™, IBM Informix™, SQLite™, and Sybase™.

Compatibility guides for writing portable SQL are provided in the *Genero Business Development Language User Guide*.

- Database drivers specific to each supported database vendor are provided as pre-linked shared libraries.
- At runtime, the DVM generates the appropriate SQL commands to be used with the target database server.

A Genero application can connect to different database servers simultaneously by issuing simple `CONNECT TO` and `SET CONNECT` instructions.

Genero Web Services

Genero support for Web Services is built in to Genero BDL.

Genero developers do not need to learn intricate programming to use Web Services; they can use simple embedded commands in BDL to create and use Web services.

Web Services work by answering requests for information and returning data in structured XML documents. As XML is simple text and Web Services can be invoked via the hypertext transfer protocol (HTTP), it does not matter what platform runs the Web Service.

Typically, web services use the Simple Object Access Protocol (SOAP) or Representational State Transfer (REST) protocols to define the communication and structure of messages.

Genero Desktop Client

The Genero Desktop Client is a front end client that displays application screens natively on your Windows, Linux, or MAC OS. The GDC also allows you to run your application through the GAS, yet deliver it locally using the GDC. Shortcuts can store the information necessary to start an application.

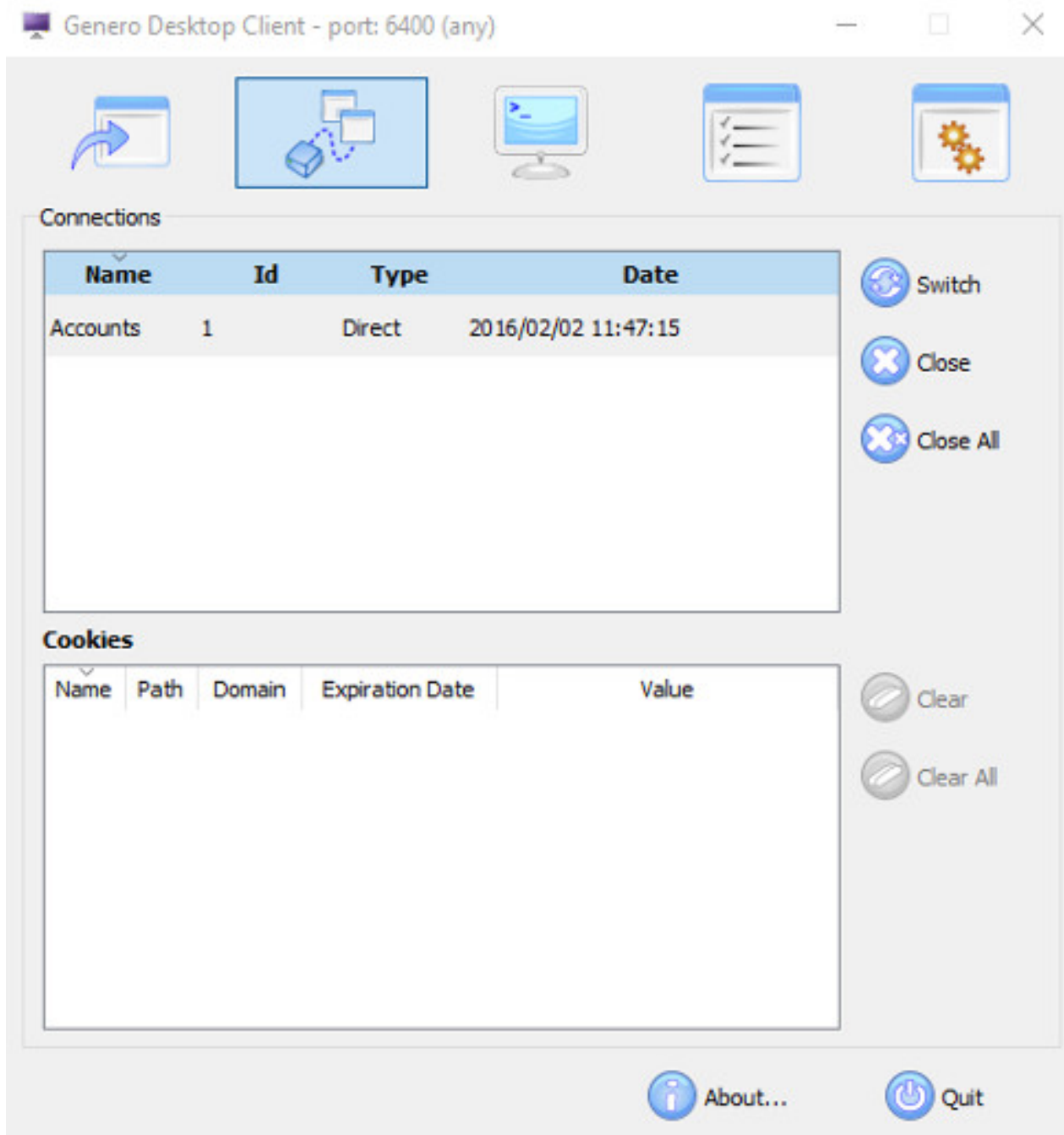


Figure 3: Genero Desktop Client

Genero Application Server

The Genero Application Server (GAS) is an engine that allows you to develop Genero applications for delivery to Genero front-end clients for both the desktop and the web. No changes to your Genero application source code and form program files are required; the same Genero application can be displayed in a browser or as a desktop application.

The GAS is embedded with a Web Server; it includes dispatcher and proxy processes to enable the GAS to be interfaced with a Web Server to handle requests from the Internet. For development cycle you can install the GAS locally and use the GAS's Standalone HTTP capabilities to serve the web page. But a Web server (e.g. Apache™ httpd, lighttpd, nginx, or Microsoft™ ISAPI) must be included for deployment.

The Genero Web Client component of the GAS supports development of both highly scalable database-oriented web applications, and enterprise applications that provide web and desktop interfaces to their functionality. You can customize and control how the Web Client renders the application to provide a more web-like interface when the application is displayed in a browser. This is done by the front end client software rendering the Genero applications for display in standard HTML browsers.

The GAS establishes and manages the communication between Genero front end client software (GDC or GWC) and the DVM. Communication between the front-end client and the GAS is handled by the Genero dispatcher, which routes requests from the Web Server to the correct GAS daemon. Several GAS daemons can be configured to load balance the requests.

The GAS also manages a pool of DVMs for Web Services applications.

Tour Genero Studio

Genero Studio accelerates the development and management of Genero business applications by providing tools for designing interfaces, writing and debugging programs, and teaching new developers the basics of Genero Business Development Language (BDL). Take a short tour to discover what it can do.

[Launch Genero Enterprise](#) to discover the following features and tools Genero Studio provides:



Figure 4: Genero Studio Components

- [Project Manager](#)
- [Code Editor](#)
- [Graphical Debugger](#)
- [Form Designer](#)
- [Meta-schema Manager](#)
- [Report Writer](#)
- [Business Application Modeling](#)
- [Code Analyzer](#)
- [Integrated Help](#)

Launch Genero Enterprise

Windows users: Select **Start** > **Four Js Genero Evaluation Program 3.00.xx** > **Genero Enterprise**

Linux users: Select **Desktop Menu** > **Four Js Genero Evaluation Program 3.00.xx** > **Genero Enterprise**

When Genero Enterprise is launched, the **Welcome Page** displays a list of the available Genero projects. In this tour, you will explore Genero Enterprise by working with the `Reports` project. Additional demo programs are also available for you to try, see [Running the Demos](#) on page 32. At any time you can select the **help icon**



in the Genero Enterprise toolbar to get additional help from the documentation.

Project Manager

The Project Manager helps you manage projects and their associated files. Genero Studio supports Source Code Management (SVN).

Reports is a pre-defined project that contains multiple applications and the corresponding source code files.

1. Choose the project **Reports** from the listing in the **Welcome Page**. A project (`Reports.4pw`) maps the relationships between the nodes in the Reports project. The structure of the project is displayed in the **Projects** view.
2. Expand the **Reports** structure tree in Projects view by clicking the **+** sign. The **Reports** project consists of application and library nodes that contain the files associated with the various applications in the project.

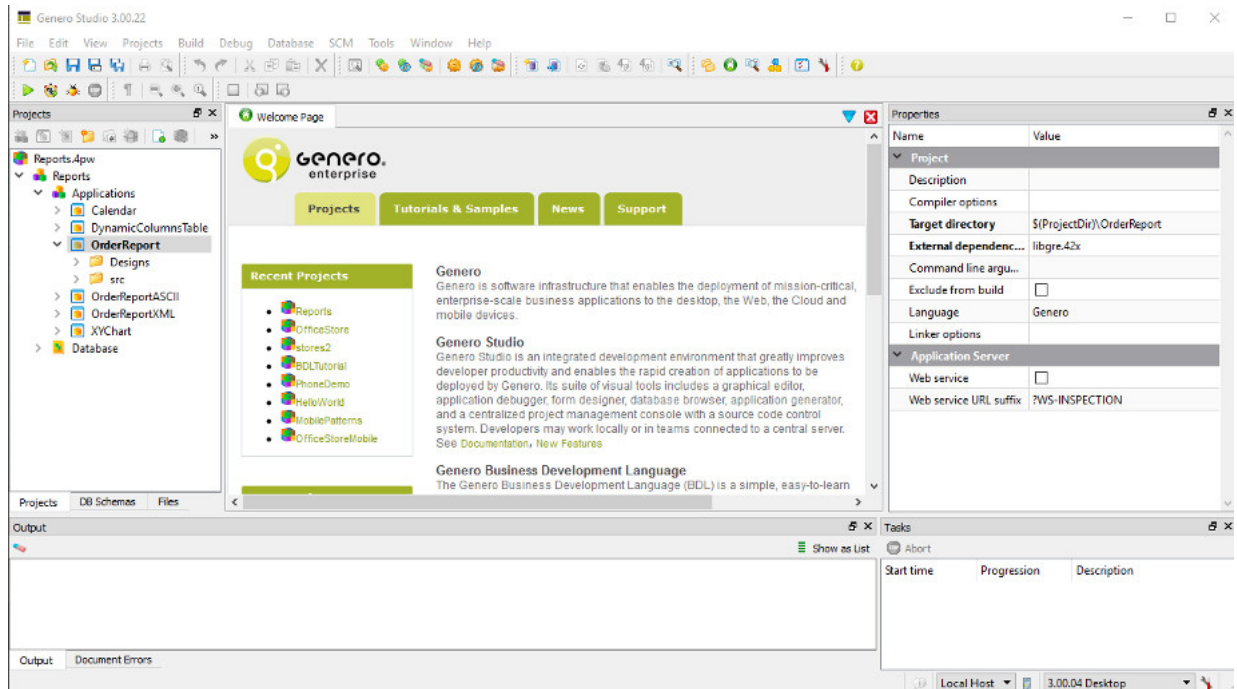


Figure 5: Genero Studio Project View

3. Expand the application **OrderReport**. Within this project, expand the listings. The project contains two virtual folders:
 - **Designs** - contains the report design documents associated with the application
 - **src** - contains the Genero BDL files and other files associated with the application

Code Editor

Code Editor is a structured, graphical editor with real time Genero syntax checking, code completion, code template management, and more. It is a programming-oriented editor designed primarily for editing Genero BDL source code, but it can handle any kind of text or languages.

1. From your expanded **Reports** project, double-click the file **OrderReport.4gl**. The **Code Editor** displays the file for editing, and the toolbar now has additional icons specific to Code Editor. If you wish, you can choose **Window > Workspaces > Document** to switch your view to one designed for document editing.

The Code Structure view on the right displays the structure of the file, listing the functions and other components. Syntax errors are marked in the editing window and listed in the Output panel (Document Errors). Select a function in the **Structure** panel to display the corresponding code in the editing window.

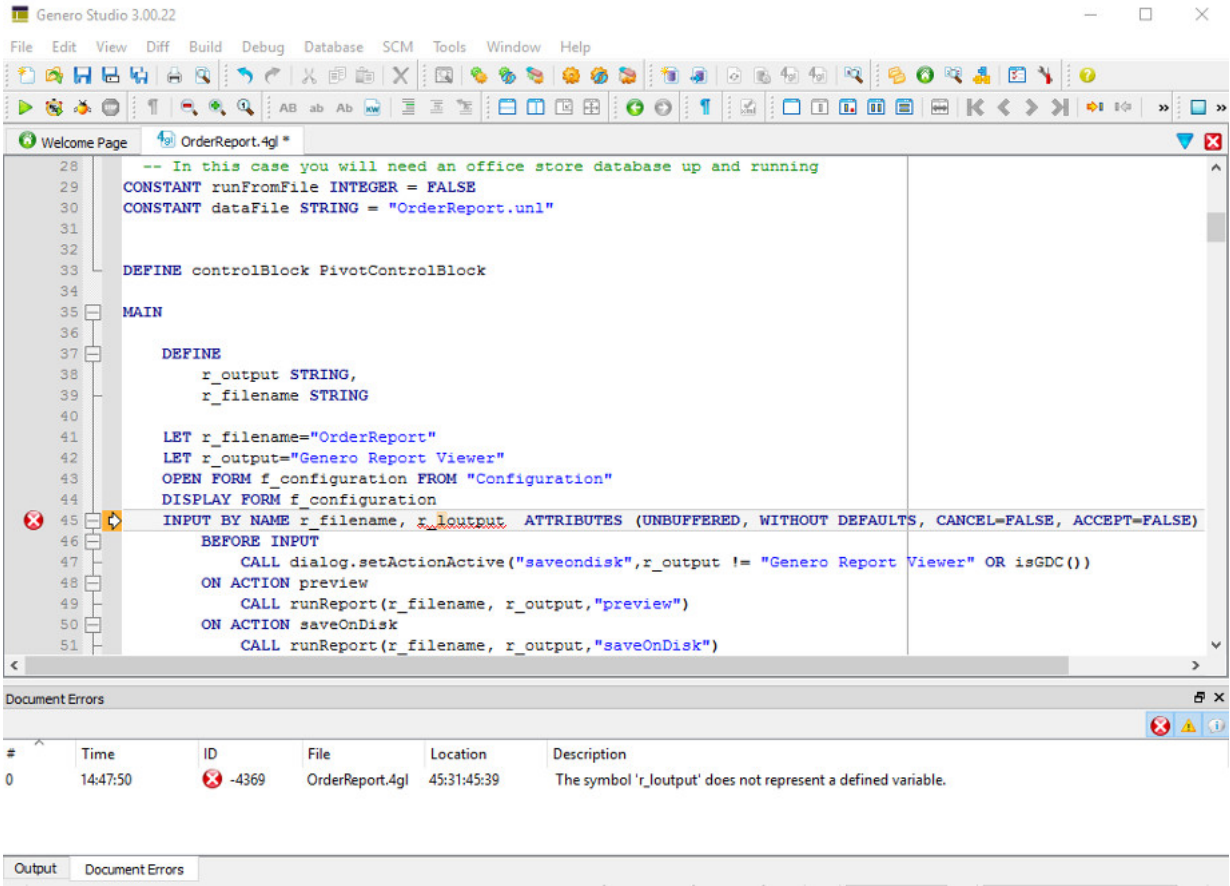


Figure 6: Genero Studio Code Editor

2. Select **File > Close OrderReport.4gl** to close the Code Editor.

Tip: If you changed your **Workspace** view, use the Studio main menu option **Window > Workspaces > Normal** to change it back.

Graphical Debugger

The Graphical Debugger is used to monitor the execution of an application, stopping at chosen points to examine the application's behavior, or to test different scenarios.

Breakpoints are set on program lines in the .4gl program file in **Studio Code Editor**, using the right-click menu:

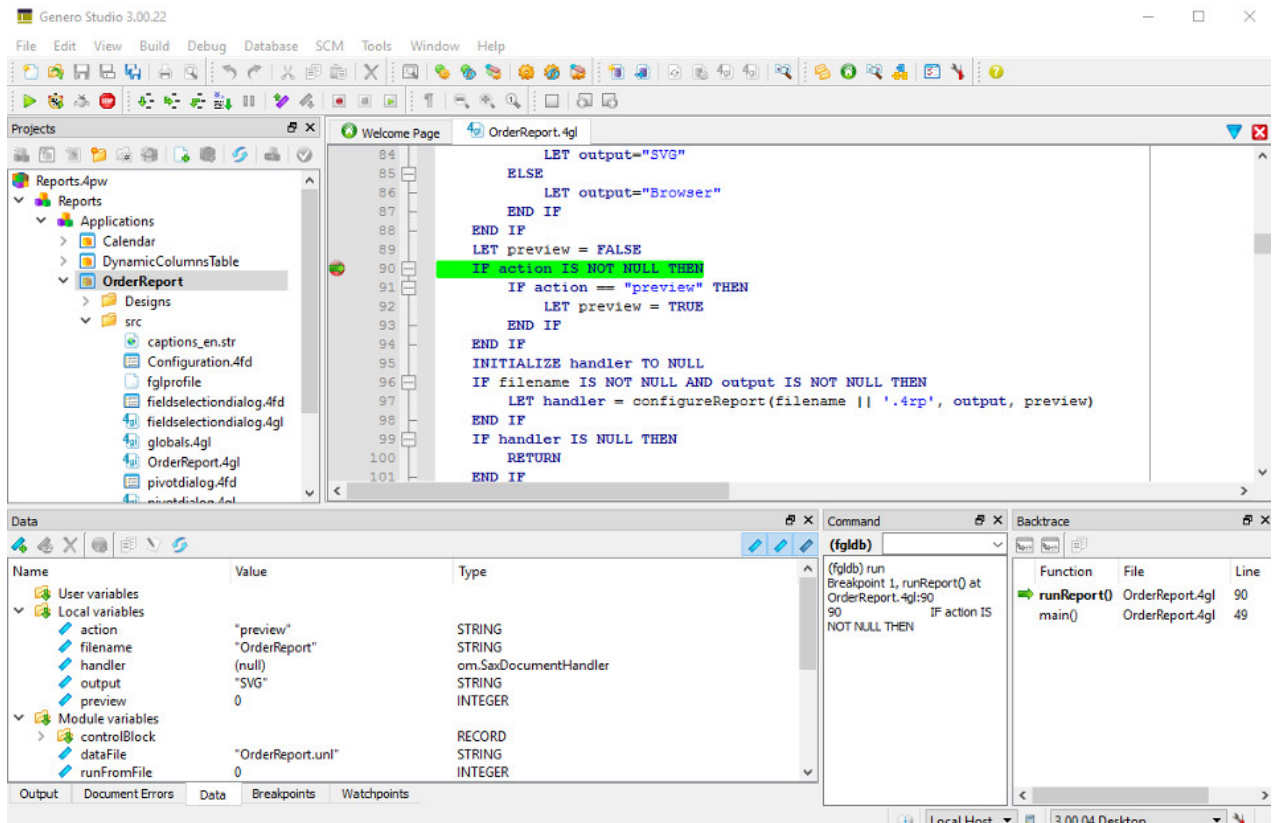


Figure 7: Genero Studio Graphical Debugger

The **Debug** menu has commands to control the execution of the program during debugging. The program variables and their current values are displayed in the **Data** tab of the **Output** view. Additional tabs allow you to manually enter commands and use watchpoints, for example.

Form Designer

Form Designer is a drag-and-drop visual editor that supports the creation, editing, and layout of Genero user interfaces. The Form Designer is integrated with a **Meta-schema Manager** to simplify the creation and modification of interfaces that are database-aware.

1. Double-click `Configuration.4fd` in the **OrderReport src** folder to open the form definition file in Form Designer.

The form definition consists of containers and widgets that are displayed in the form and listed in the **Form Structure** view. The application displays the form to the user. High-level BDL statements in the Genero BDL file `OrderReport.4gl` handle the user's input on the form. This form allows the **OrderReport** application user to select a report and its output format.

2. Select **Build > Preview** from the Studio menu to see how the form will display to the user. Select **Yes to All** if the **New connection** window is displayed. Close your previewed form.
3. The properties of a selected form object is listed in the Properties view. **Select the RadioGroup** (as shown) to see its properties in the **Properties** view.

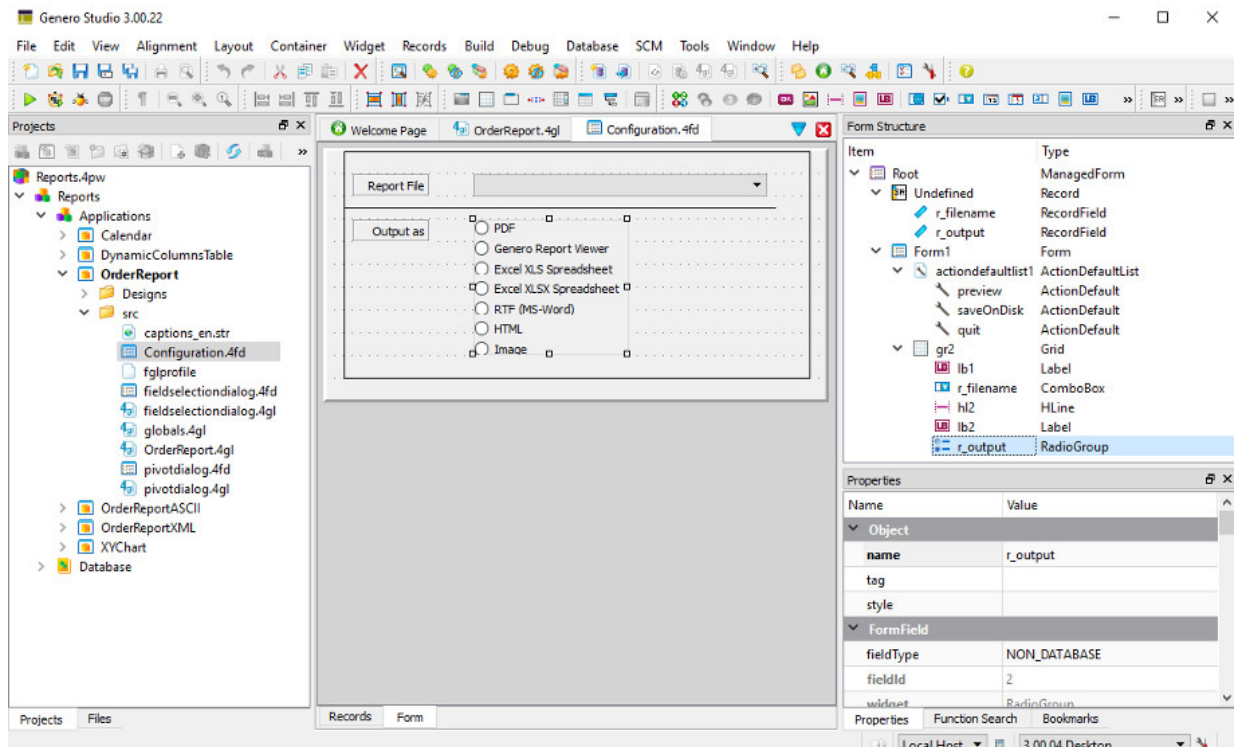


Figure 8: The Form Designer

4. Form Designer is a drag-and-drop visual editor. Select the **grid** around the form items and drag the bottom to make it larger. Then, re-arrange the form objects as you wish.
5. Select **Build > Preview** to see how your changes appear to the user.
6. Select **File > Close** from the Studio menu to close the form and exit Form Designer. Do **not** save the changed form.

Note: To examine a more complex form, select the **OfficeStore** project from the **Welcome Page**. Expand the project nodes **Office Store Demo, Entities** and double-click **OrderForm.4fdm**. Close the Form Designer before continuing with the next exercise.

Meta-schema Manager

A meta-schema is an XML file that is a central repository of the Database Metadata; the information about the tables, columns, and relations of a relational database. It also provides the appropriate default values when using the tables, columns, or relations in Genero Studio Forms or Studio-generated applications. The **officestore.4db** schema is used in the **Reports** project. Double-click on **officestore.4db** to explore it in the Meta-schema Manager.

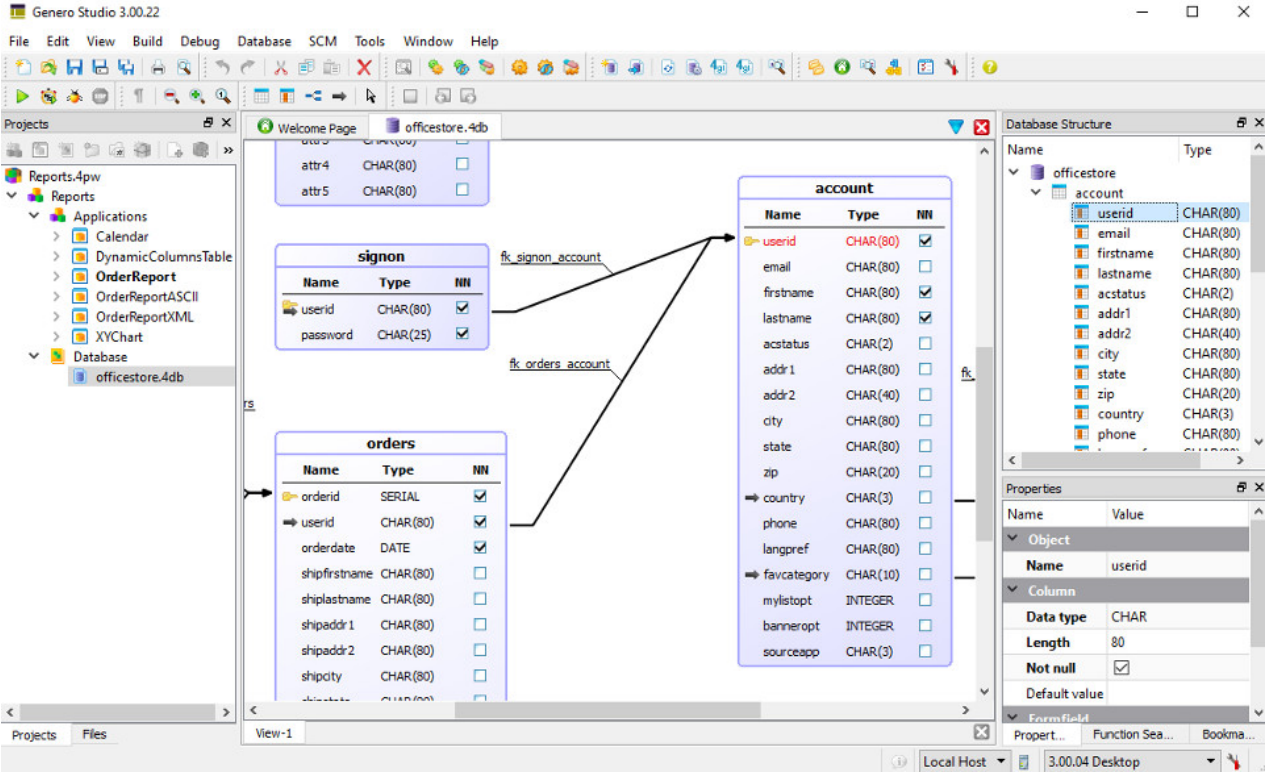


Figure 9: The Meta-schema Manager

Report Writer

The Genero Report Writer is a powerful tool for the creation of business reports for mission-critical applications. The Report Writer includes:

- Genero Report Designer (GRD) for graphical report layouts.
- Genero Report Engine (GRE) to process the raw data.
- Genero Report Viewer (GRV) to render the report.

1. Double-click the **OrderReport.4rp** in the **Designs** folder in the **Projects** view to open one of the report designs used by the Order Report application.

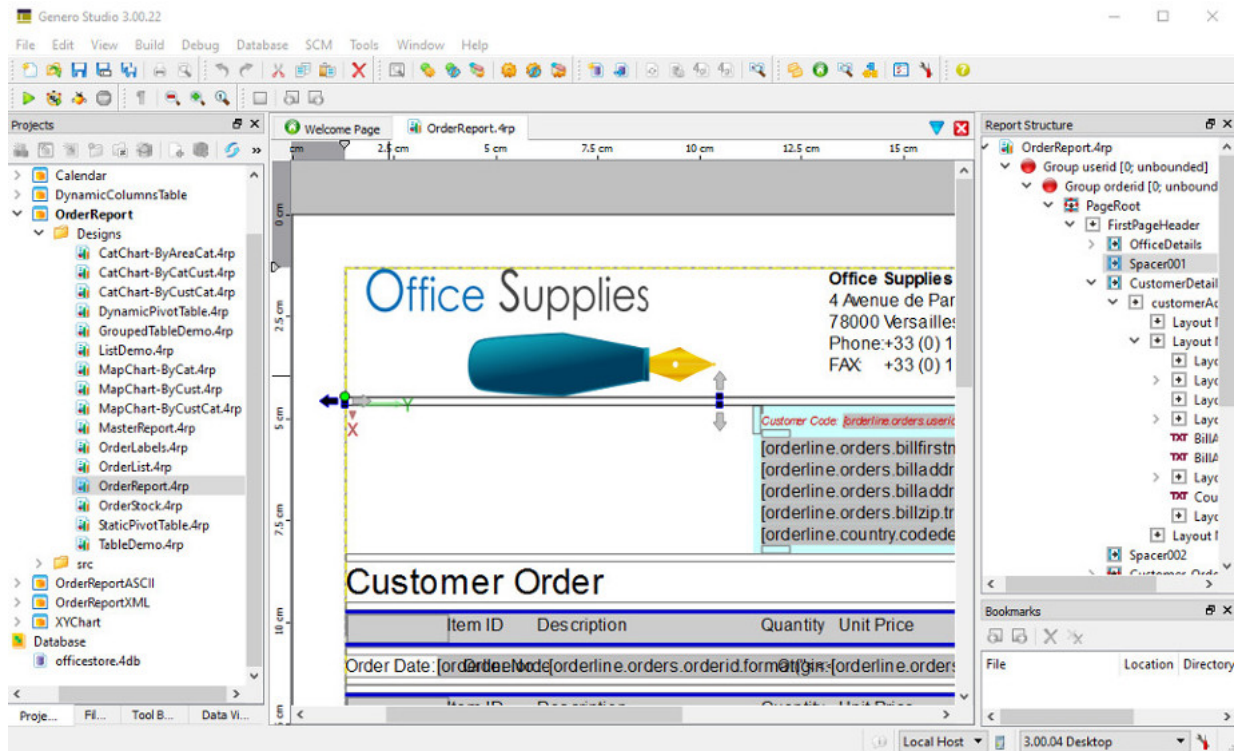
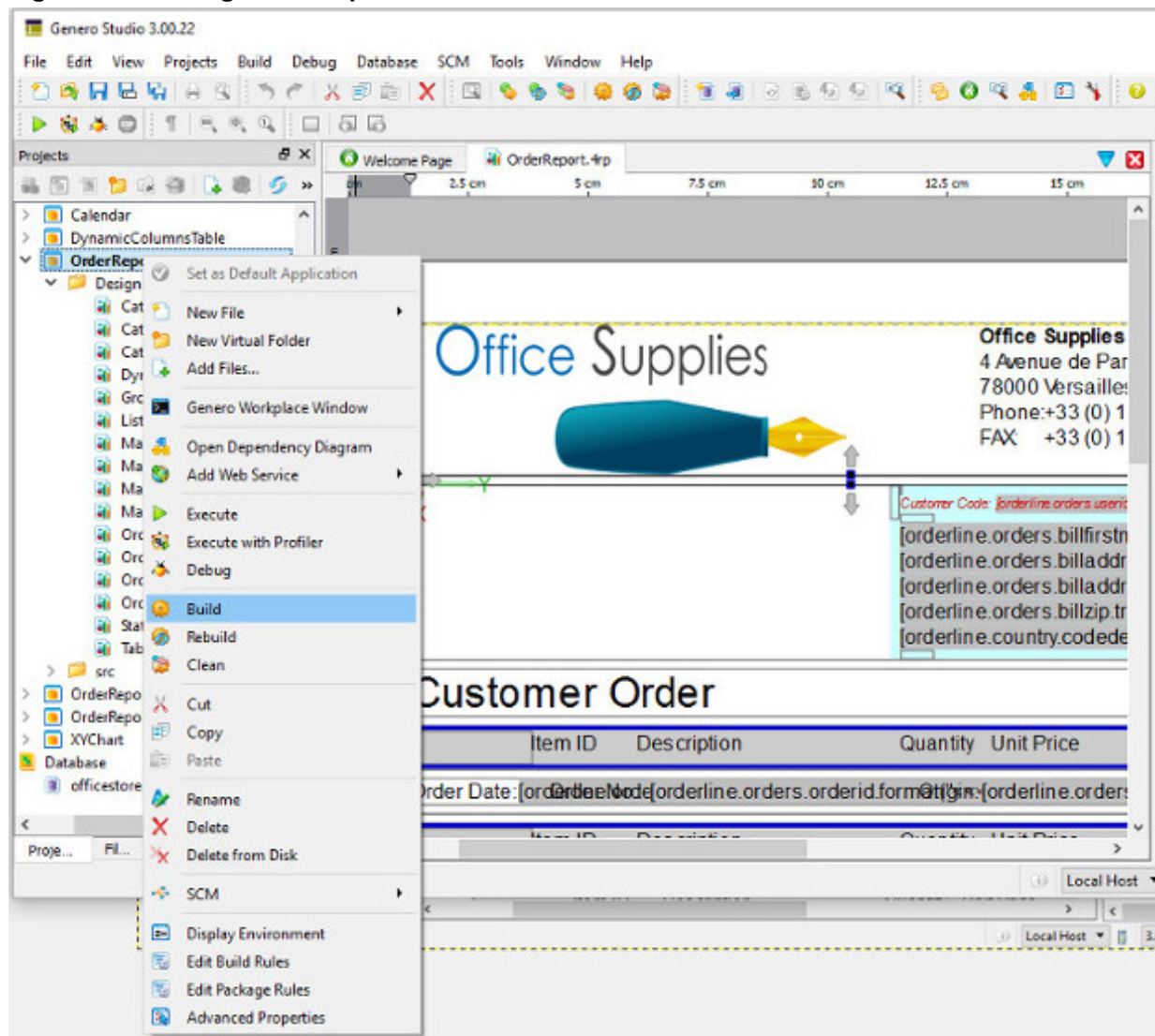


Figure 10: Genero Report Designer

2. The **Toolbox** tab allows you to drag containers and other objects onto the report design, and the **Data View** tab allows you to add the report data. The **Properties** view lists the property values for a selected report object. The report's tree structure is listed in the **Report Structure** view.
3. Close the **OrderReport.4rp** document.

4. Compile and link the **OrderReport** application - right-click the node in the **Projects** view and select **Build** from the context menu. The output from the **Build** operation displays in the **Output** tab in the bottom panel.

Figure 11: Building Order Report



5. Run the program - right-click the **Order Report** application node and select **Execute** from the context menu.
6. The Genero Desktop Client is called automatically to display the **Configuration** form, the application's user interface:
 - Select Order Report from the **Report File** dropdown box
 - Select SVG as the **Output**
 - Select **Preview**

The Order Report is displayed in the Genero Report Viewer.

Business Application Modeling

Business Application Modeling is the process of modeling applications. The Business Application Modeler (BAM) is the set of modeling tools that allow you to graphically model your application. The BAM works with Studio components, such as Meta Schema Manager, Business Application Diagram, Form Designer, Report Designer, Report Data designer and Web Services data designer to:

- Create a design model (.4ba) of your application.
- Generate forms (application screens) based on the database meta-schema.
- Generate the source code for a Genero application, using the design model and generated forms as input.

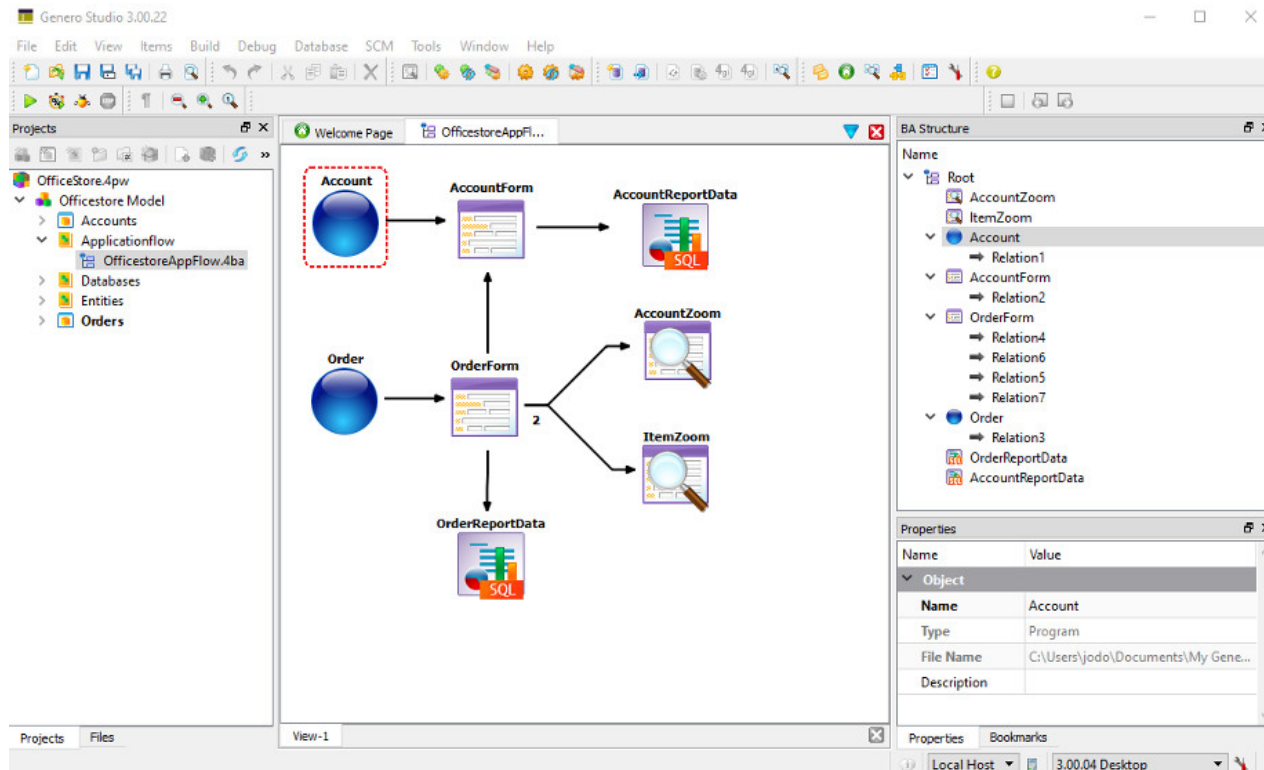


Figure 12: The Business Application Modeler (BAM)

The Business Application Modeler permits you to outline the behavior of your application in diagrams. It manages the Genero BDL language code for you. The code is updated as you declare new behavior in the diagrams.

The generated program:

- Allows the user to search a database table, add a new database row, update a database row, or delete a database row.
- Allows the user to retrieve a value for a form field from a list displayed in a pop-up window.
- Populates the items for a ComboBox using values from a database table.
- Handles the master-detail relationships between tables, such as between orders and order items as shown in the example.

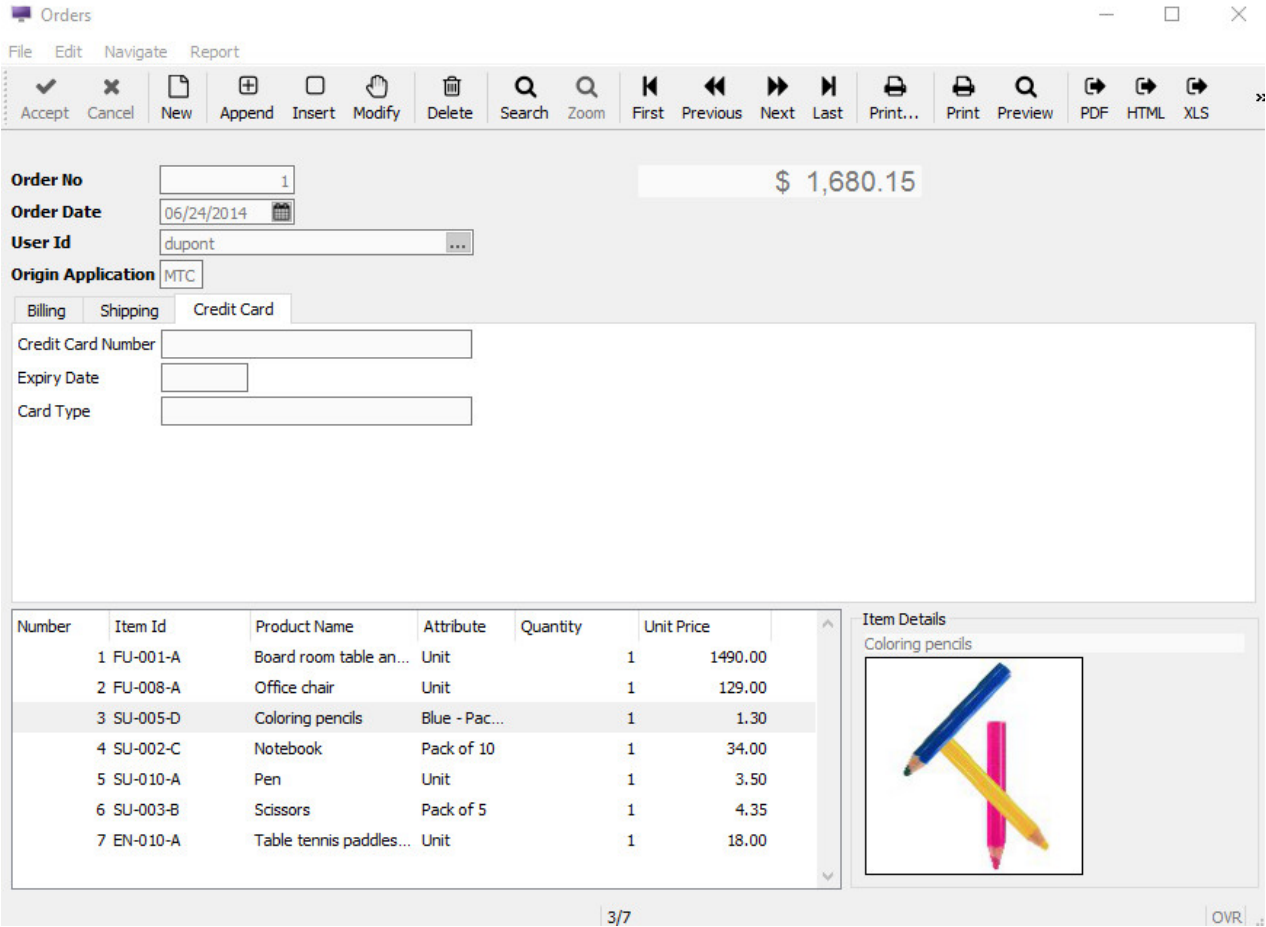


Figure 13: Example Order Detail Application Screen

Code Analyzer

The Code Analyzer reverse engineers your existing applications for component relationships and function call sequence. It generates diagrams, providing you with an overview of your application. Select the link between components to display a list of function calls.

Dependency Diagram - displays the complex relationships between application and its components, the dependencies between the various objects. To display the diagram, right-click the **Application** node or the **Project** node, and select **Open Dependency Diagram** from the menu of options:

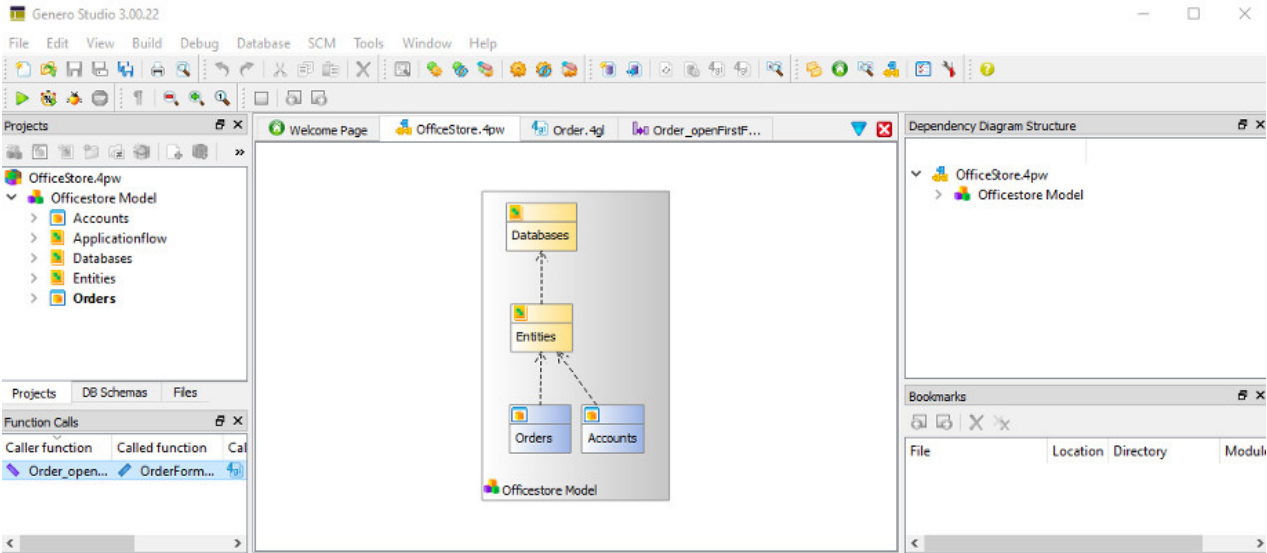


Figure 14: Example Dependency Diagram of Officestore Application

Sequence Diagram - displays the application's BDL functions and how they interact with each other, which functions call and/or are called by other functions. To display the diagram for a function from an open Genero .4gl source code file - right-click the **function** name and select **Open Sequence Diagram** from the context menu:

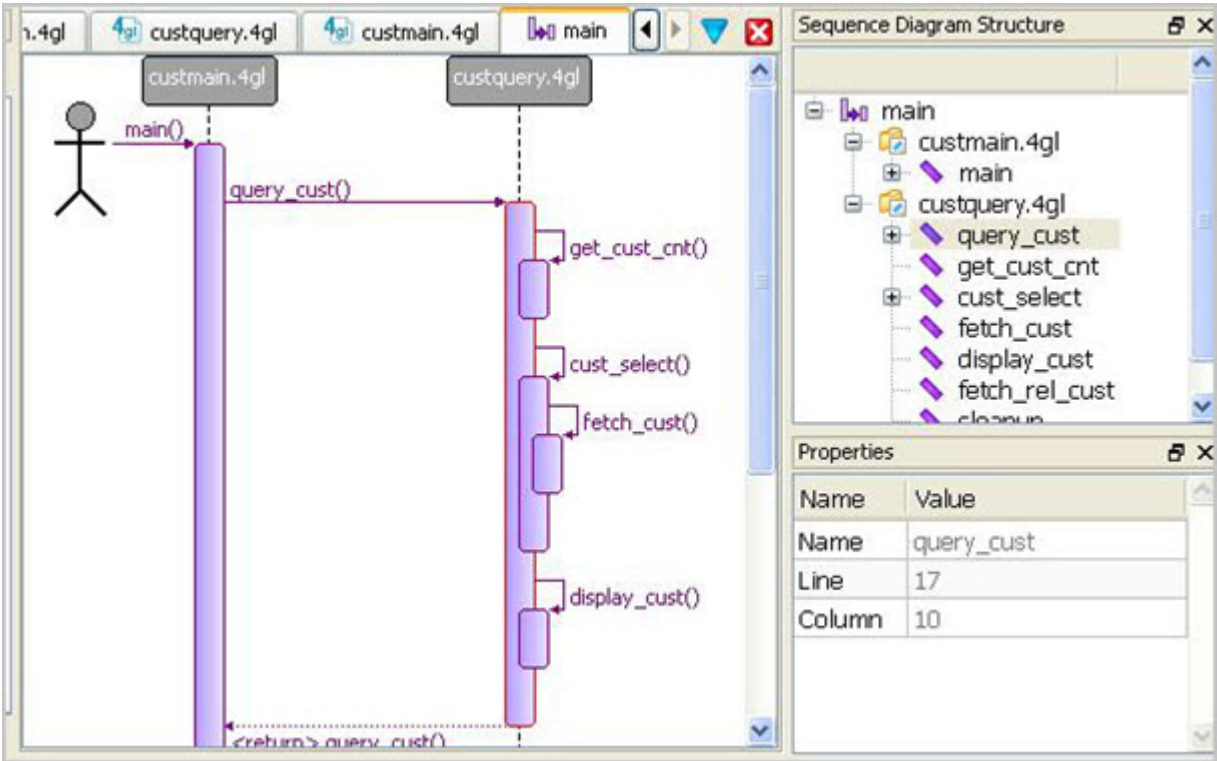


Figure 15: Example Sequence Diagram of Officestore Application

Integrated Help

Online help is available from within the **Genero Enterprise Program > Help** menu.

Tip: You can also get context help for a feature or tool you are working with by pressing the **F1** key, or selecting the **help icon**



in the Genero Enterprise toolbar.

You can also access Genero Studio Help from the **Four Js Genero Evaluation** program list:

- Windows users: Select **Start** > **Four Js Genero Evaluation Program 3.00.xx** > **Genero Studio Help**
- Linux users: Select **Desktop Menu** > **Four Js Genero Evaluation Program 3.00.xx** > **Genero Studio Help**

The complete Genero documentation sets is displayed in the **Contents** tree in the left panel of the help window as shown, which you can browse or search using the **Index** or **Search** tools.

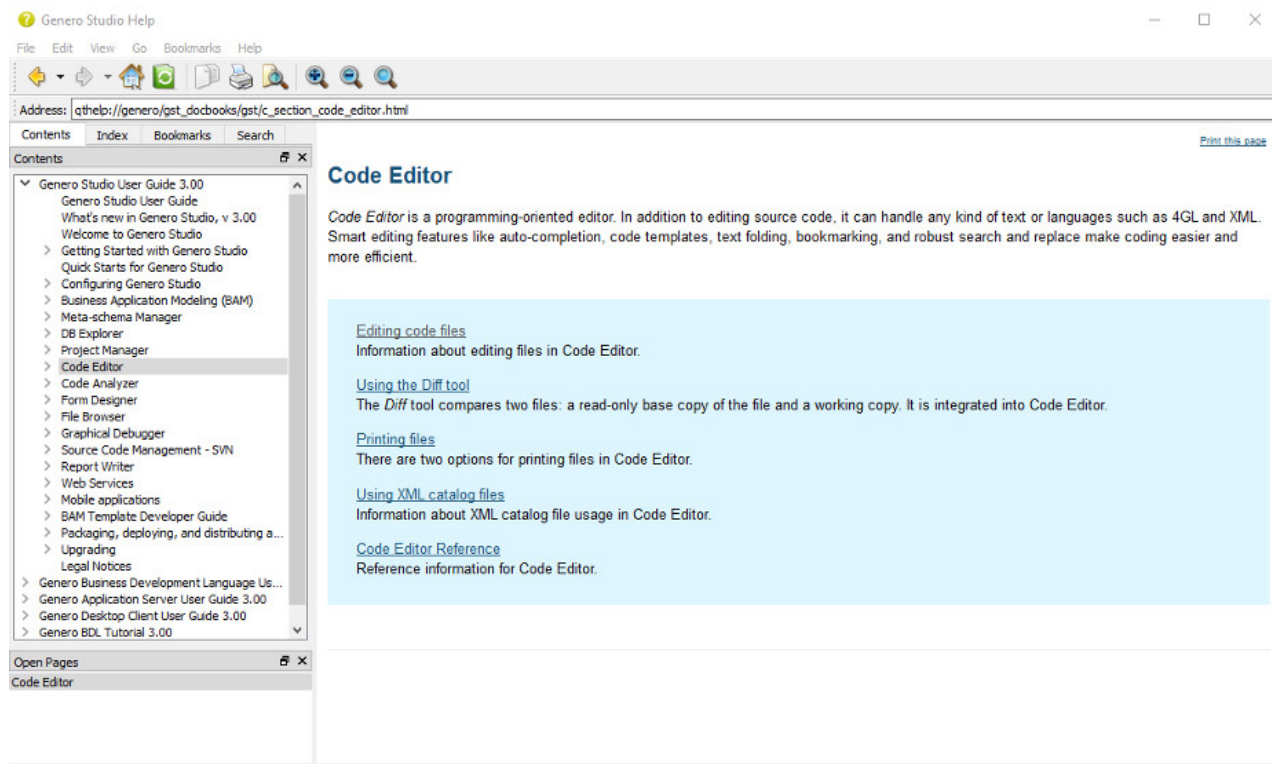


Figure 16: Genero Studio Help

Running the Demos

Genero Studio includes demo programs that illustrate some typical business applications.

To run a demo:

1. Launch Genero Studio.
2. From the **Welcome Page**, select the **Tutorials & Samples** tab and the demo that you want to run.
3. Right-click the application node, and select **Execute**.

Continue reading for a brief explanation of two of the demos.

Office Store demo

The Office Store demo displays orders and accounts from the included Office Store database.

Click on the link to open the **OfficeStore** demo. Under the **Officestore Model** group, execute the **Orders** application.

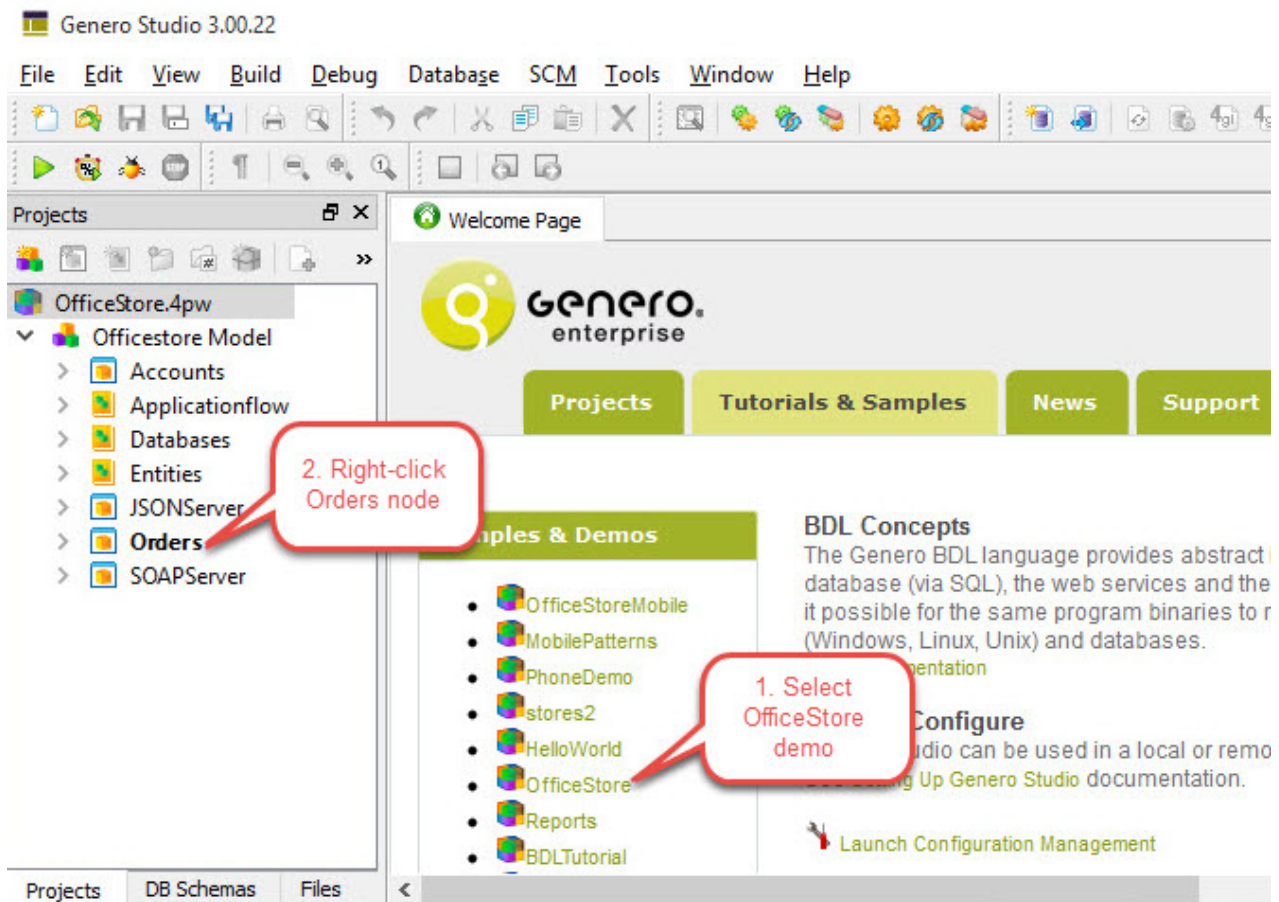


Figure 17: Execute the Orders application

You can then interact with the Office Store database by using the Orders application.

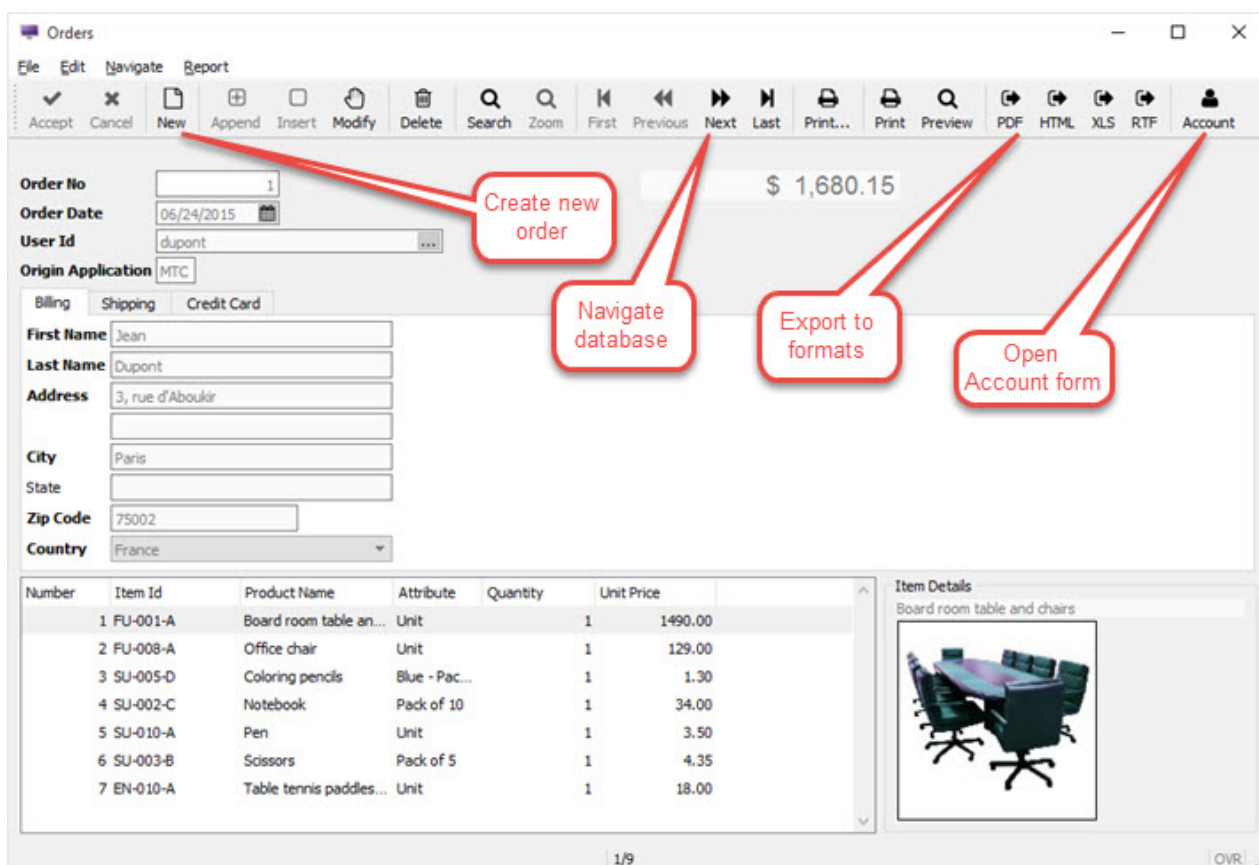


Figure 18: Using the Orders application

Reports demo

The Reports demo provides a sample reporting application with various reports design documents. For example, you can view the report data in a list or a chart.

Open the **Reports** demo. Under the **Reports > Applications** group, execute the **OrderReport** application.

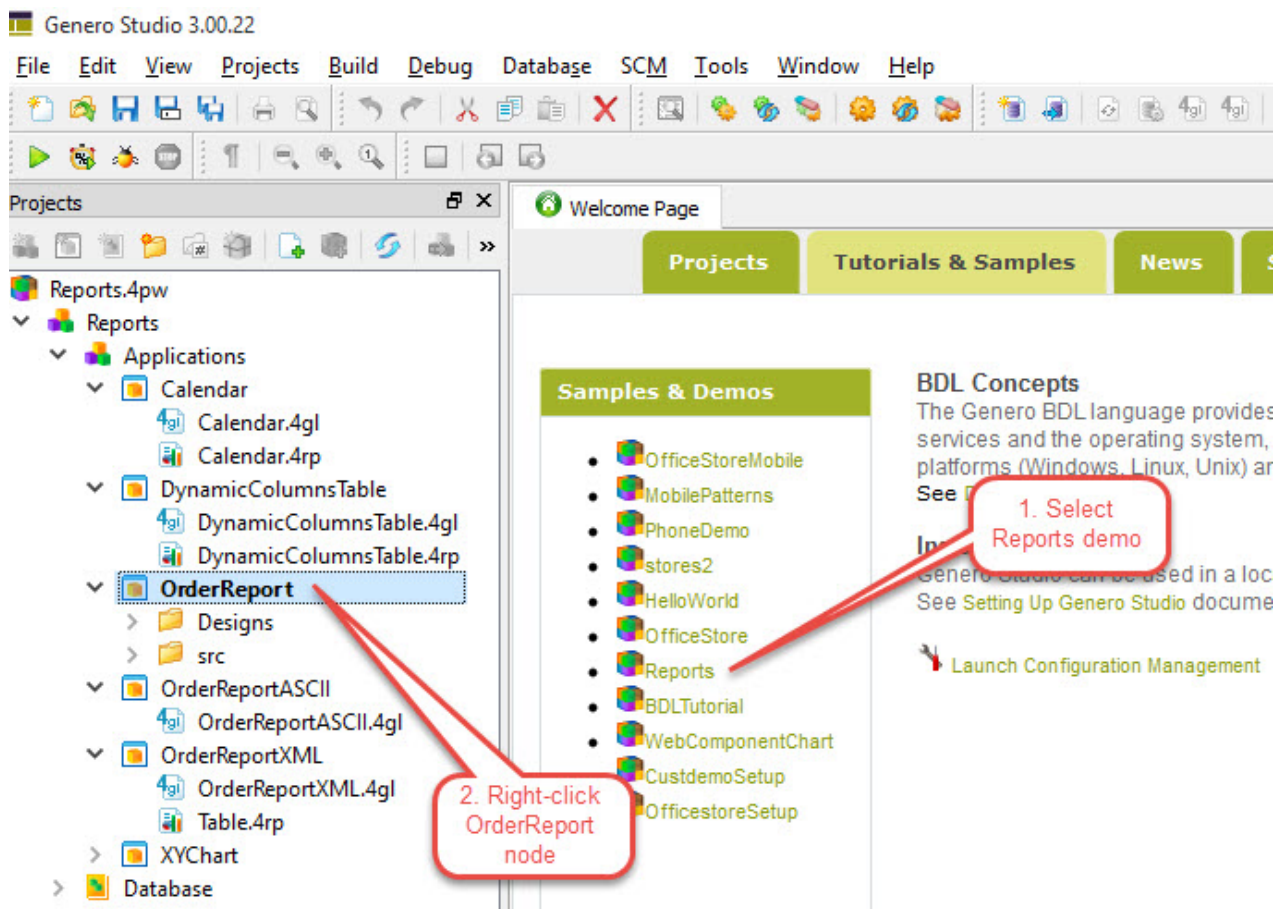


Figure 19: Execute the OrderReport application

From the application, you can select the report file and the type of output. To start, we recommend you use the default values of **OrderReport** and **Genero Report Viewer**. Click **Preview** to see the report.

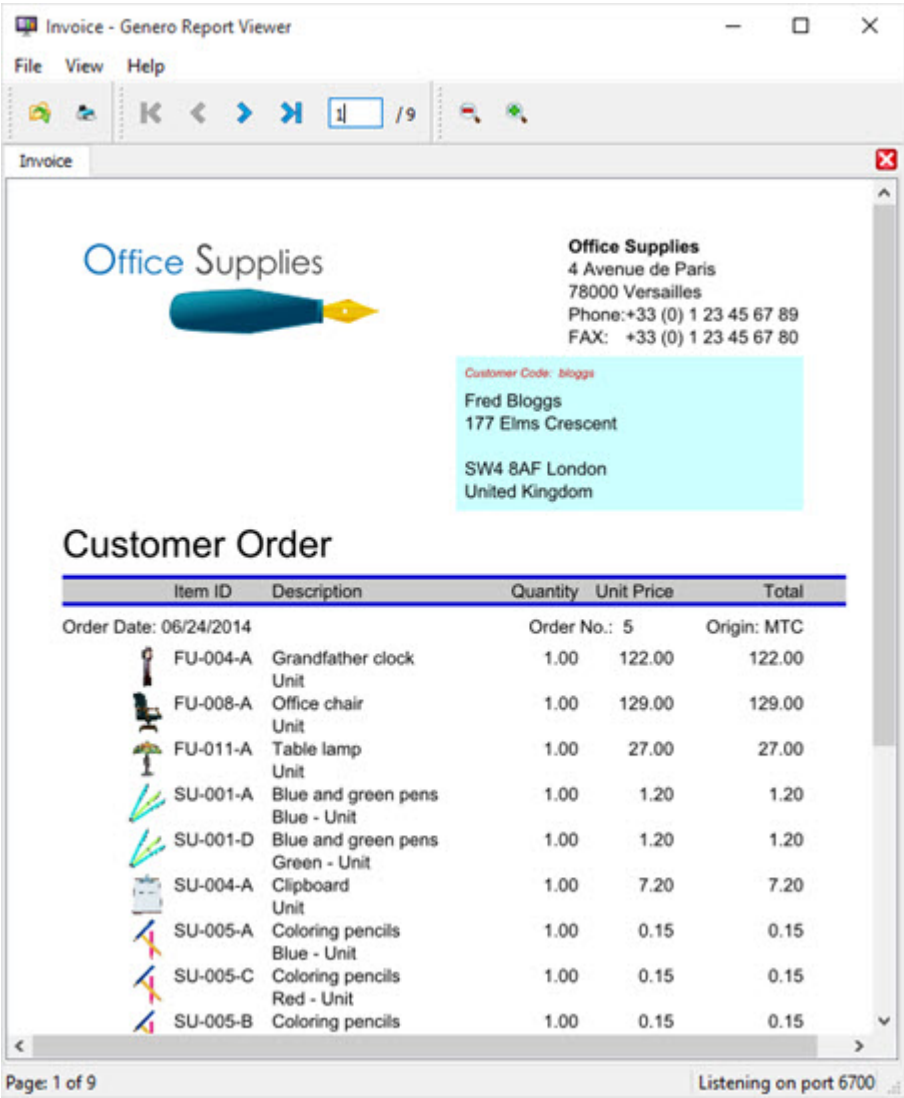


Figure 20: The OrderReport report

Switching Genero Clients

Genero Studio installs with a set of configurations that launch your application using various Genero front ends.

Front-end options

Your options include:

Desktop

When you select the Desktop configuration, the application displays in the Genero Desktop Client. This configuration works out-of-the-box.

Web

When you select the Web configuration, the application displays in the Genero Web Client for JavaScript. This configuration works out-of-the-box.

Web (deprecated)

When you select the Web (deprecated) configuration, the application displays in the Genero

Web Client for HTML5. This configuration works out-of-the-box.

Tip: The Genero Web Client for HTML5 is provided for legacy Genero applications, and is provided for backwards compatibility only. We recommend you use the Web configuration - and the Genero Web Client for JavaScript - for your web applications.

Android (ARM)

When you select the Android (ARM) configuration, the application displays on an Android device.

Important: This configuration will NOT work out-of-the-box. You must complete additional configuration steps and provide an Android device. See [Configure Genero Mobile for Android](#) on page 118.

Android (x86)

When you select the Android (x86) configuration, the application displays on an Android virtual device.

Important: This configuration will NOT work out-of-the-box. You must complete additional configuration steps. See [Configure Genero Mobile for Android](#) on page 118.

iOS Dev Client

When you select the iOS Dev Client configuration, the application displays on an iOS device.

Important: This configuration will NOT work out-of-the-box. You must complete additional configuration steps and install the Genero Mobile Development Client on your iOS device. See [Display to the Genero Mobile Development Client](#) on page 138.

Select your configuration

Select a configuration from the combobox located in the lower right corner of Genero Studio, then run your application.

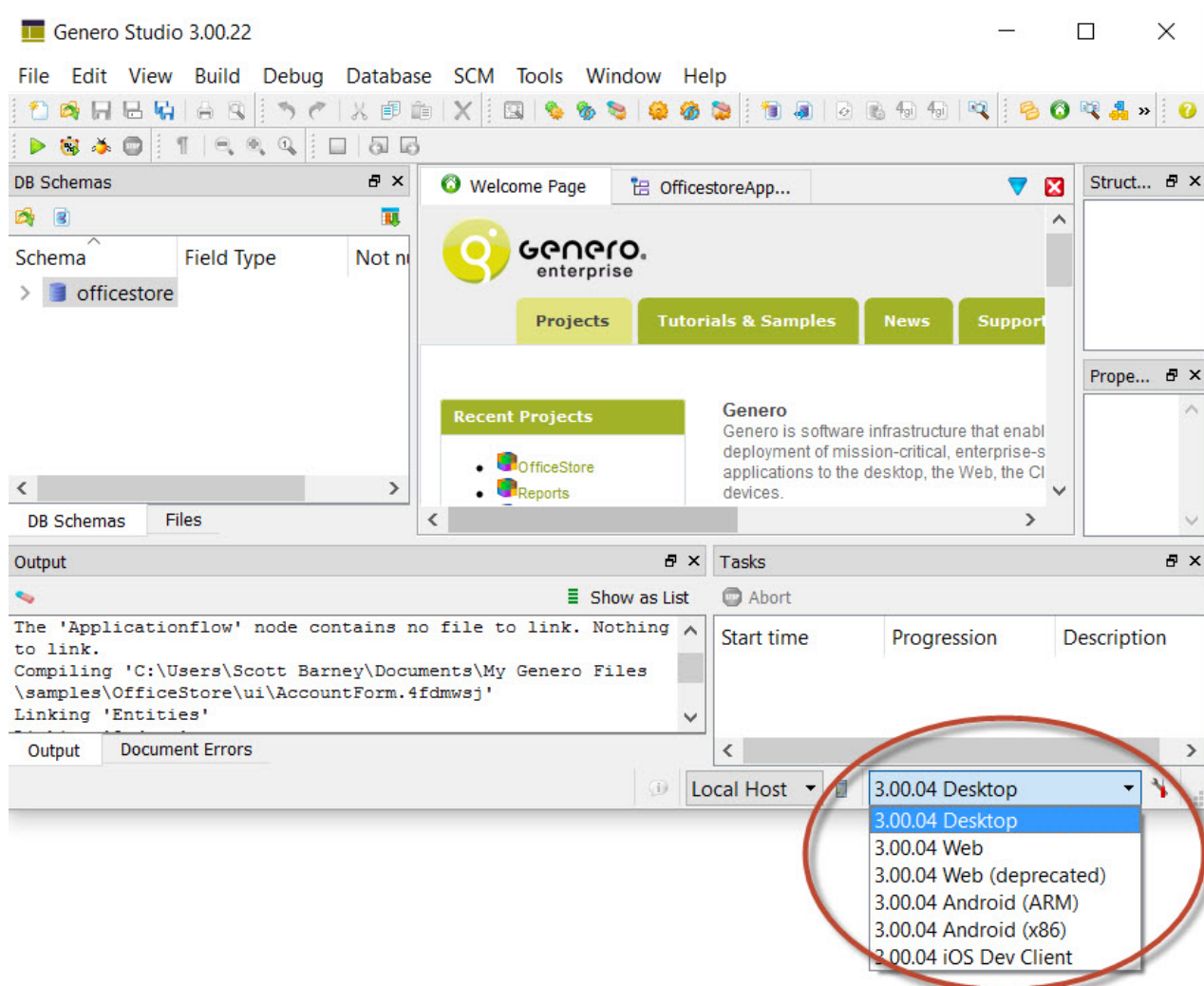


Figure 21: Selecting the configuration for the desired client

Creating with Quick Starts

A quick start provides you with simple step-by-step instructions for completing specific tasks.

Quick starts are located in sections where they have supporting topics. Here is a list of all the quick starts that can be found in the *Genero Studio User Guide*.

Getting Started with Genero Studio

[Quick Start: Tour of Genero Studio](#) on page 41

Business Application Modeling (BAM)

[Quick Start: Generate an application](#) on page 176

[Quick Start: Generating a mobile app](#) on page 182

Project Manager

[Quick Start: Create a project](#) on page 340

Form Designer

[Quick Start: Creating a first form](#) on page 407

BAM Template Developer Guide

[Quick Start: Customizing templates](#) on page 930

Finding more information

Genero Studio includes a comprehensive documentation suite to help you understand and use the product.

All Genero documentation is installed with Genero Studio. [Documentation is also available on our web site.](#)

The [Four Js Training Portal](#) provides access to our instructor-led training calendar and our online training videos. The training videos are targeted to programmers who are new to Genero, as well as to more experienced programmers who need to catch up with Genero and its latest features.

Welcome to Genero Studio

Genero Studio provides a graphical integrated development environment for developing and managing Genero applications.

Genero Studio is dedicated to Genero users, to help you quickly and easily:

- Work locally or across remote resources, with multiple defined configurations. See [Configuring Genero Studio](#) on page 115.
- Design, create and maintain database meta-schemas. See [Meta-schema Manager](#) on page 288.
- Design a user interface that interacts with a database. See [Creating the user interface](#) on page 407.
- Create application forms based on database tables. See [Form Designer](#) on page 406.
- Create application code that uses the correct syntax. See [Code Editor](#) on page 374.
- Create graphical reports. See [Report Writer](#) on page 551.
- Organize projects. See [Project Manager](#) on page 340.
- Compile, run, profile, and debug Genero applications. See [Building and linking programs](#) on page 343 and [Graphical Debugger](#) on page 502.
- Use version control to manage files. See [Source Code Management - SVN](#) on page 529.
- Automatically generate the logic and source code for a database application. See [Business Application Modeling \(BAM\)](#) on page 176.
- Analyze the code and generate function diagrams for existing applications. See [Code Analyzer](#) on page 402.

Getting Started with Genero Studio

- [gst_section_gst_tour.ditamap](#)
- [The Genero Studio framework](#) on page 69
- [Learning to use Genero Studio](#) on page 103

Quick Start: Tour of Genero Studio

Use this tour to quickly become familiar with Genero Studio.

This tour assumes that you have recently installed Genero Studio and have not changed the default configurations. If configurations have been changed, the results of some steps may be different than documented.

Note: If you are using Genero Mobile, see the *Configuring Genero Mobile for development* section in the *Genero Mobile Developer Guide* first. This will assist you in configuring a display client (iOS or Android). You can then run the **OfficeStoreMobile** project instead of **OfficeStore** to explore Genero Studio.

To begin, launch Genero Studio. From the **Welcome Page, Projects** tab, select the **OfficeStore** project. This opens the **OfficeStore** project file in the project view.

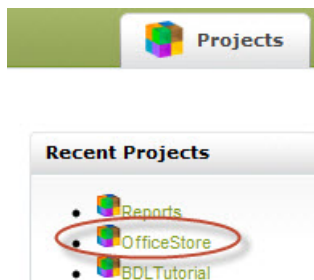


Figure 22: OfficeStore sample project

Run the OfficeStore demo

Follow these steps to run the OfficeStore demo using the Genero Desktop Client (GDC).

1. Select the Desktop configuration.

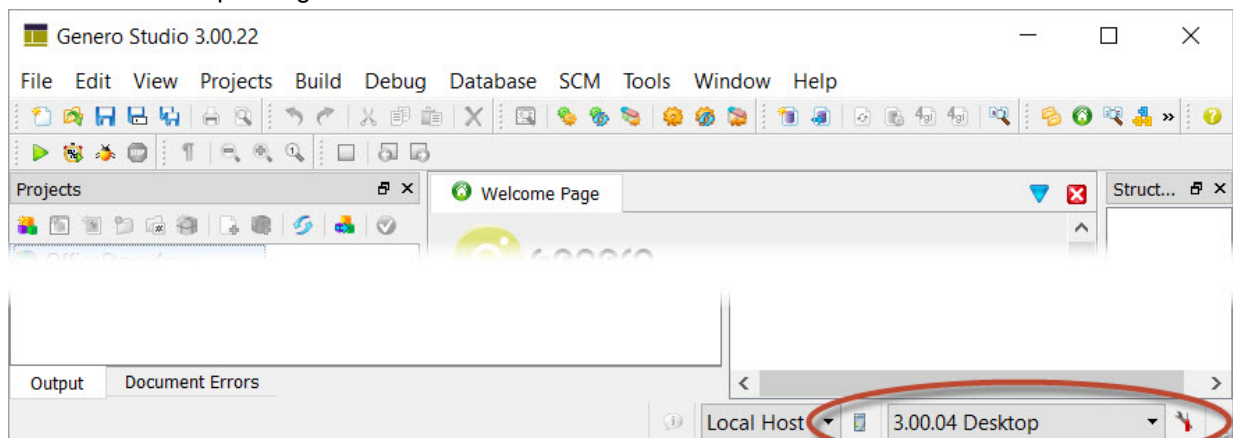


Figure 23: Selecting the Desktop configuration.

2. Expand the **OfficeStore** project and find the **Orders** program node. It is in bold because it is the default program in this project. Right-click on the **Orders** node and select **Execute**.

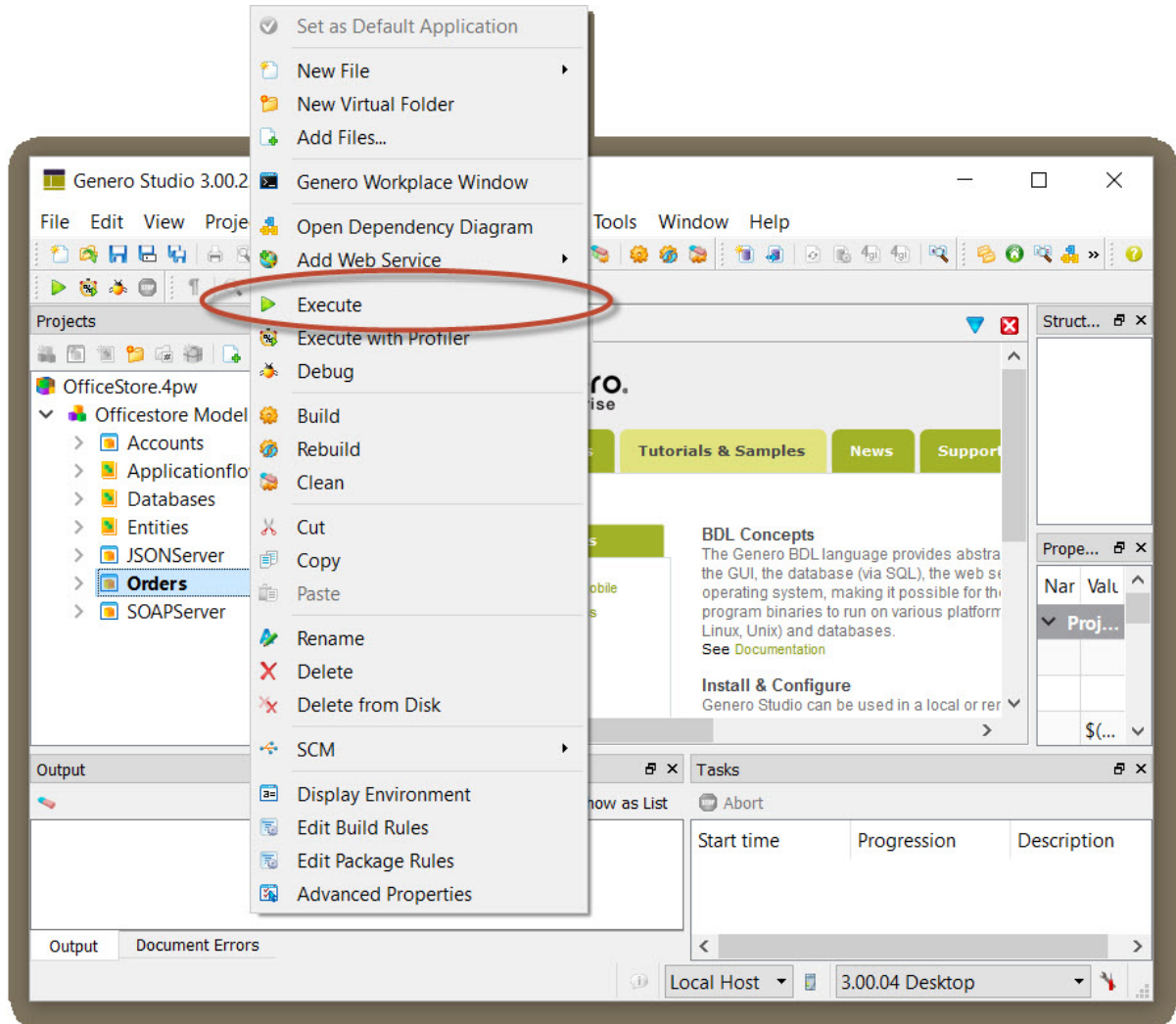


Figure 24: Executing a program

3. Navigate the **Orders** program. In these steps, browse records, add a new item to an order and modify the shipping address.
- Use the **Next** and **Previous** options in the Toolbar to browse the records.
 - Right-click on the Toolbar and deselect **Enable Text** to see the Toolbar with out the associated text.
 - Add a new item to the order. Select an item in the item list and select **Append**. In the **Number** field enter 100
 - Tab to the **Item Id** field and press the **magnifying glass icon** to bring up a list of options. Press the **Product Name** column header to sort the rows by product name. Find the product name **Basketball** and double-click to add this item to the order.
 - Tab to the **Quantity** field and enter 5.
 - Click on a different item in the list and you are prompted to save your changes. Select **Yes**. Your item has been added to the order.
 - Select the **Order Date** field and use the calendar to select a new order date.
 - Select the **Shipping** tab. In the **Address** field enter 4, Rue Beethoven.

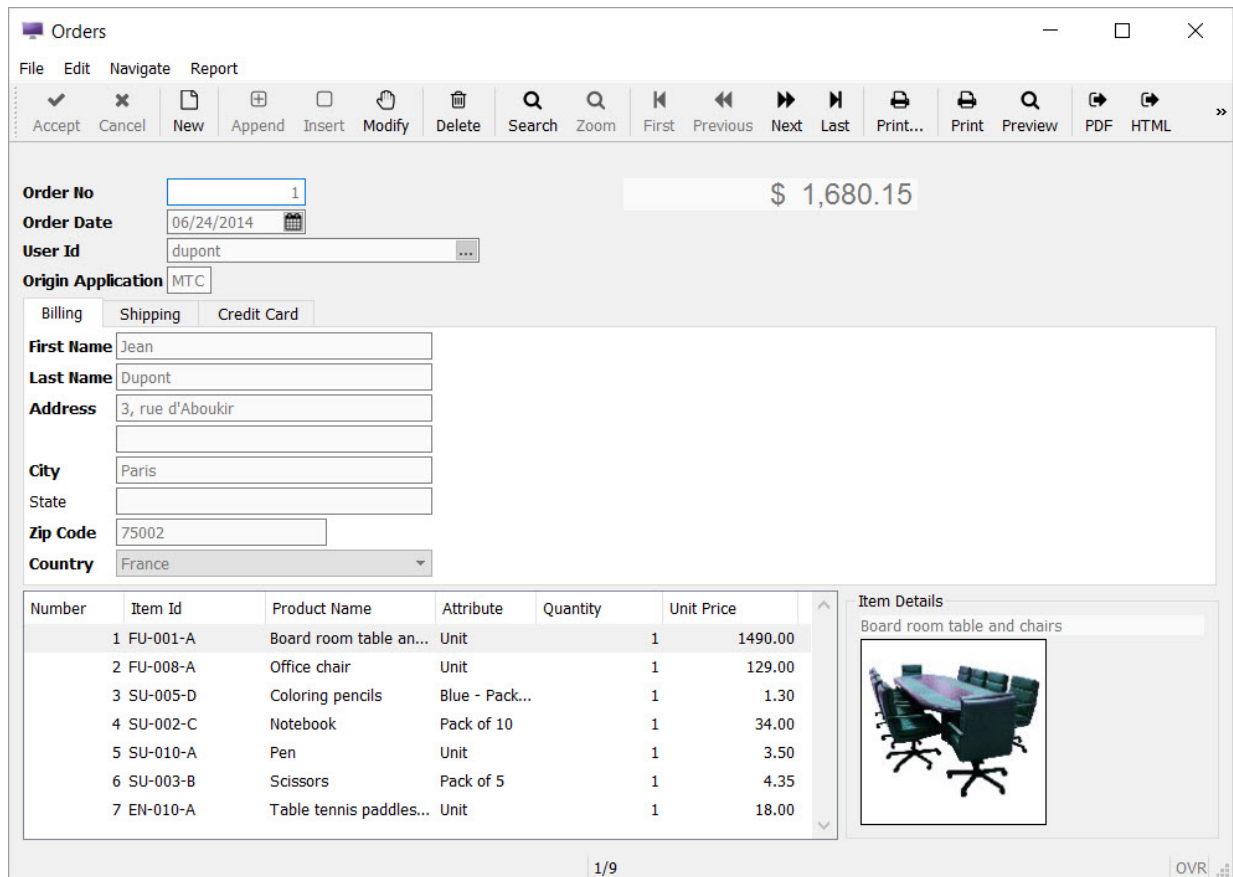


Figure 25: Program running in GDC

4. Select **File >> Exit** to exit the **Orders** program.
Run the same **Orders** program in **GWC for JavaScript**.
5. In the lower right corner, change the display configuration to **Web**.

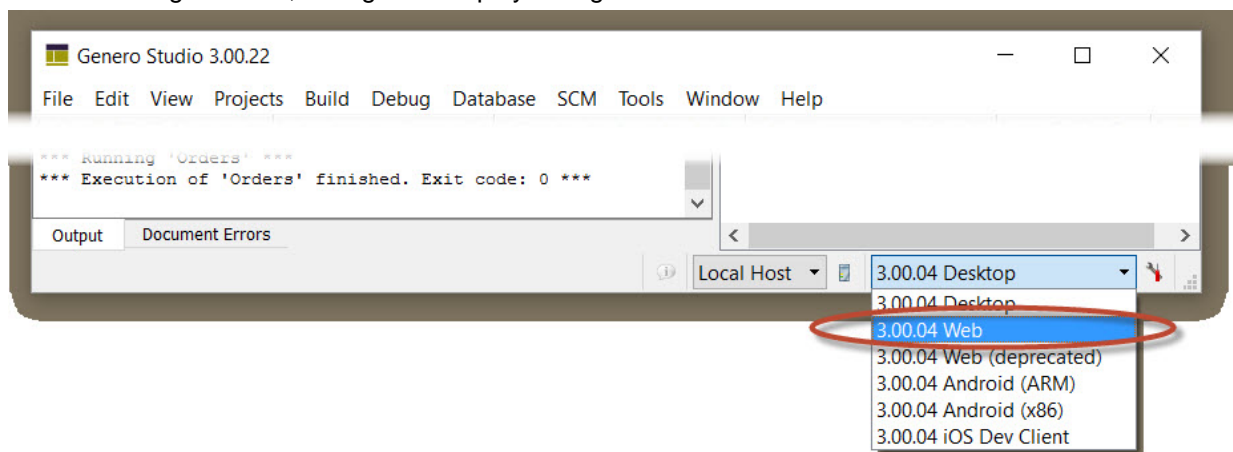


Figure 26: Change display to Web

6. Right-click on the **Orders** node and select **Execute**.

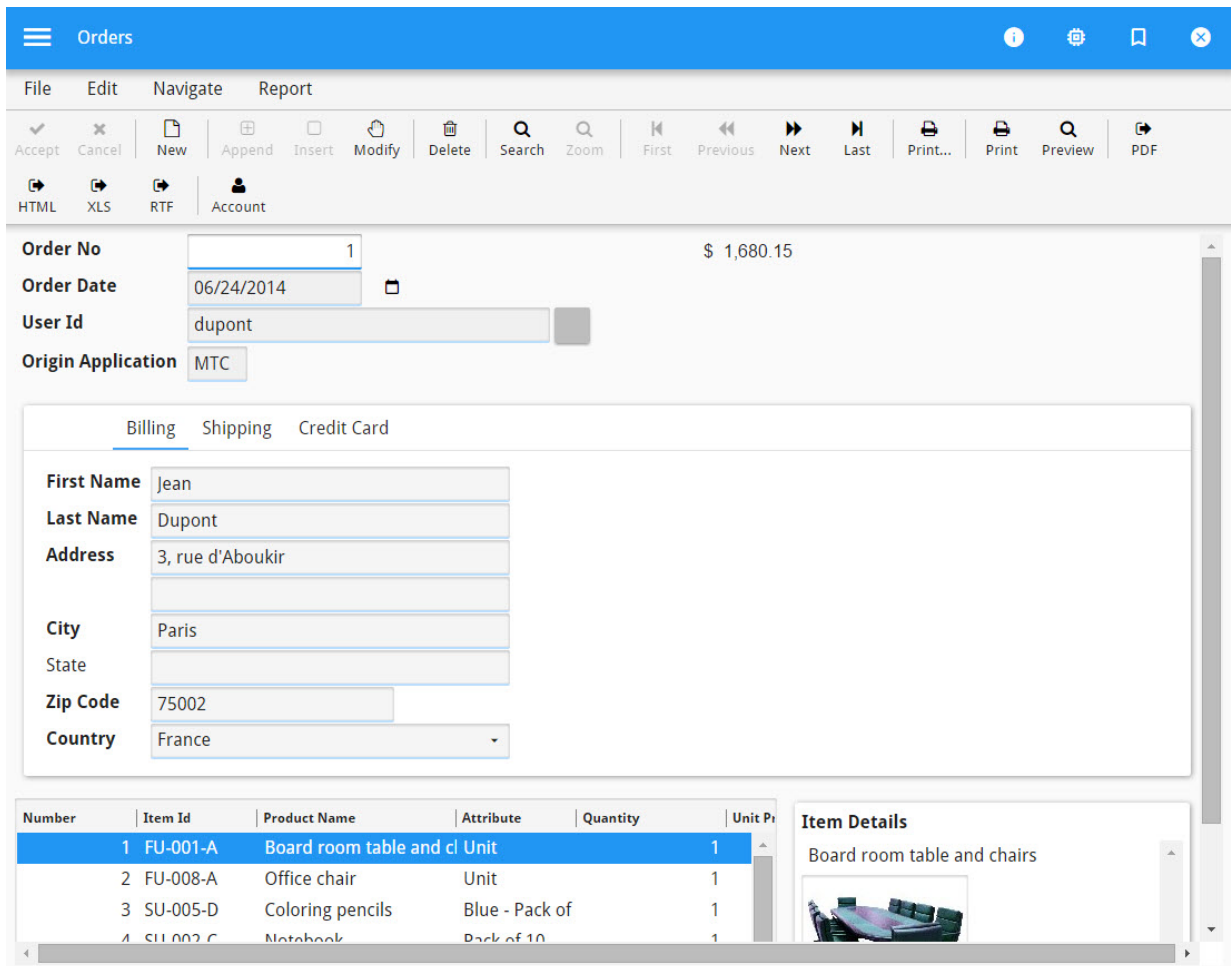


Figure 27: An application running in a browser with the default configuration for Genero Web Client

7. Navigate the **Orders** program in the Genero Web Client.
8. Select **File >> Exit** to exit the **Orders** program.

Explore the Debugger

Follow these steps to explore Debugger features.

1. Expand the **Intermediate Files** folder in the **Accounts** node and double-click `Account.4gl` to open the source in Code Editor.

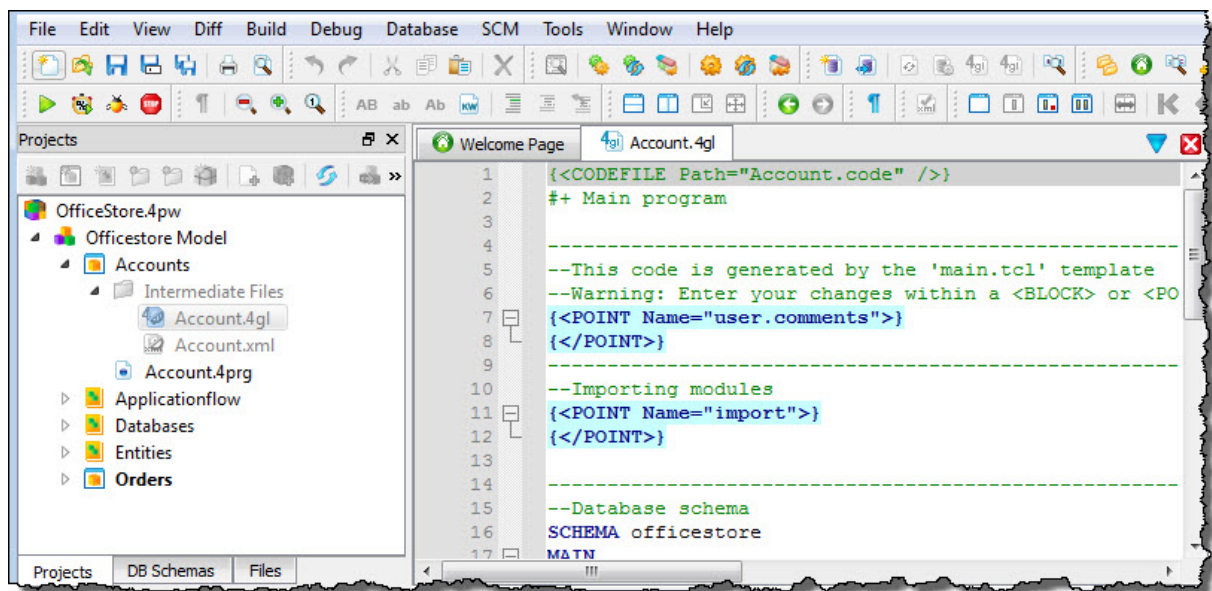


Figure 28: The Account application open in Code Editor

2. Right-click on line 33, a function call to `AccountForm_ui_uiOpenForm()` and select **Add/Delete Breakpoint**.

A red dot, the breakpoint icon, appears in the gutter adjacent to line 33.

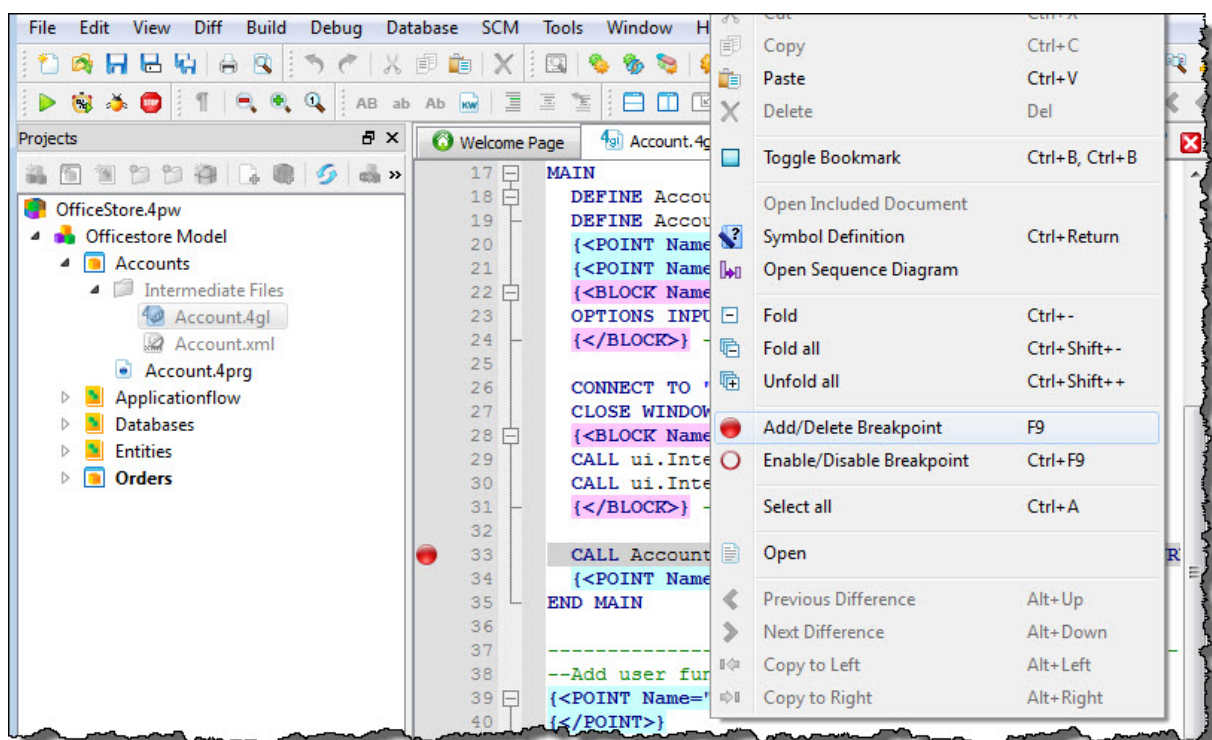
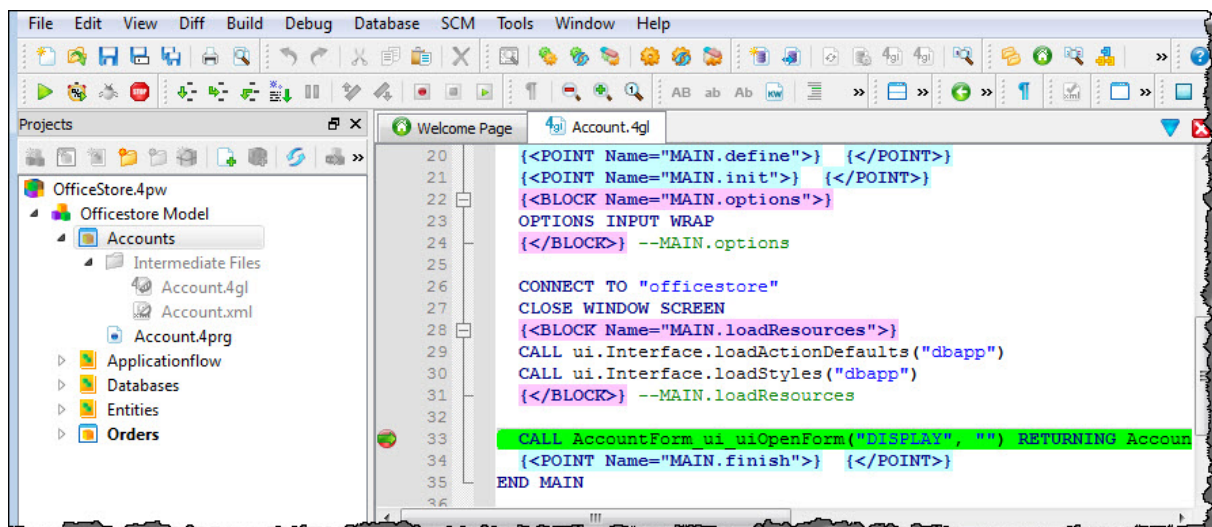


Figure 29: Adding a breakpoint

3. Right-click the Accounts application node and select **Debug** to start the application in Debug mode. When the application runs, it stops at the breakpoint, and waits.

Figure 30: Waiting at a breakpoint



4. Select **Debug >> Step in** to execute the function call and open the function module, AccountForm_ui.4gl, in Code Editor.

Execution stops inside the called function, waiting for the next Debugger instruction.

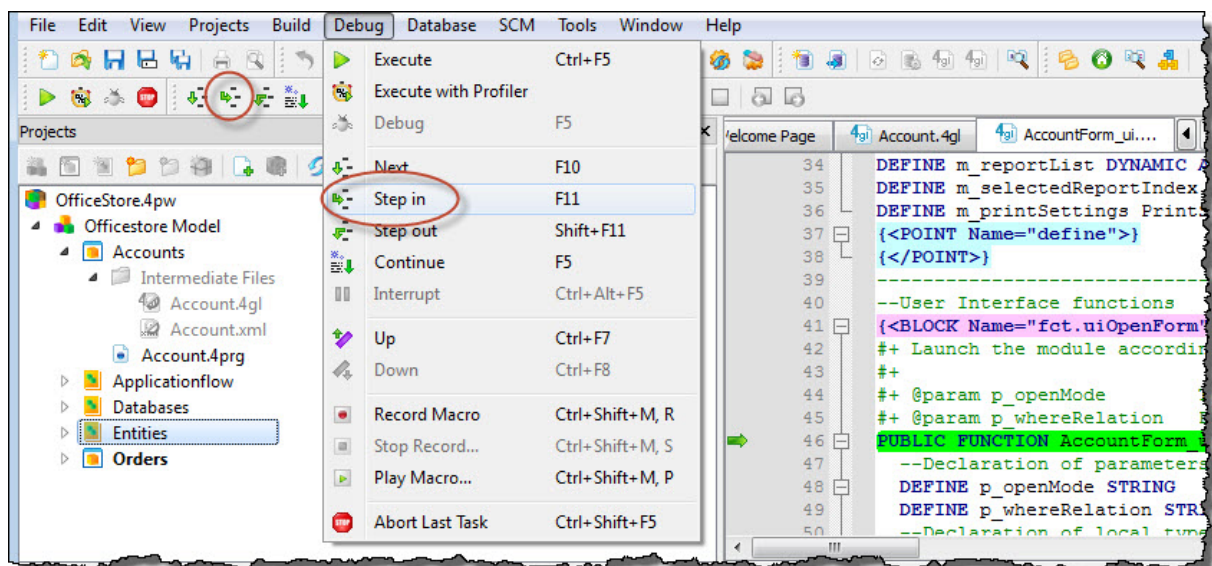


Figure 31: Stepping into an external function

5. Use the Code Structure view to quickly locate functions in AccountForm_ui.4gl for additional breakpoints:
 - a) In the Code Structure view, select the AccountForm_ui_uiDisplay() function to display the function source in Code Editor.
 - b) In Code Editor, right-click on the line containing the function header and select **Add/Delete Breakpoint**.
 - c) Repeat Sub-steps a and b to create a breakpoint for the AccountForm_ui_uiInput() function.

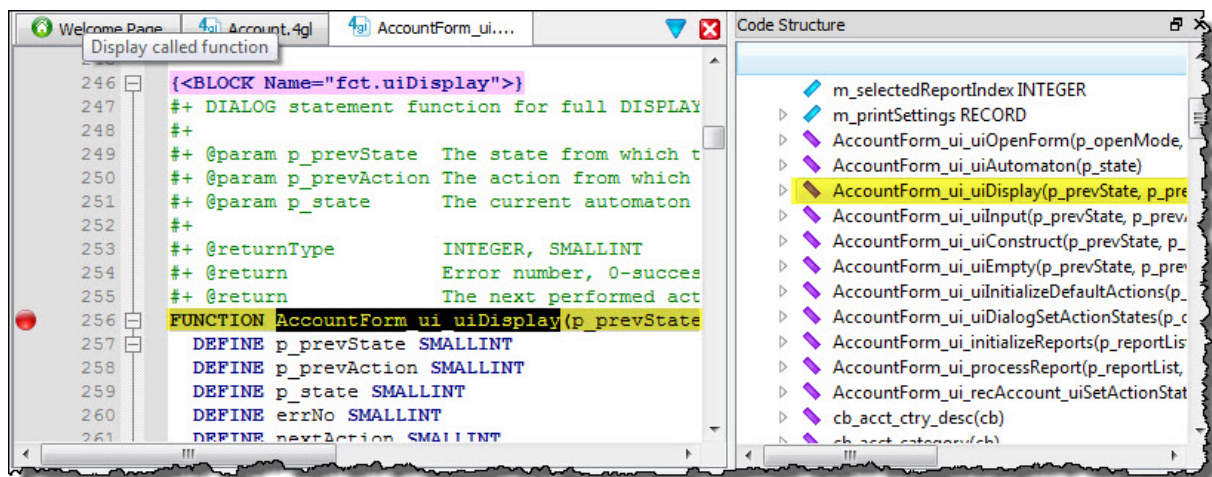


Figure 32: Navigating functions using the Code Structure view

6. In the Data view examine the values of local, module and global variables.
7. In the Data view, expand the **Local variables** folder, right-click on the variable *p_openMode*, and select **Add to watch**.

Setting a watchpoint on the *p_openMode* variable will stop program execution each time the variable value changes.

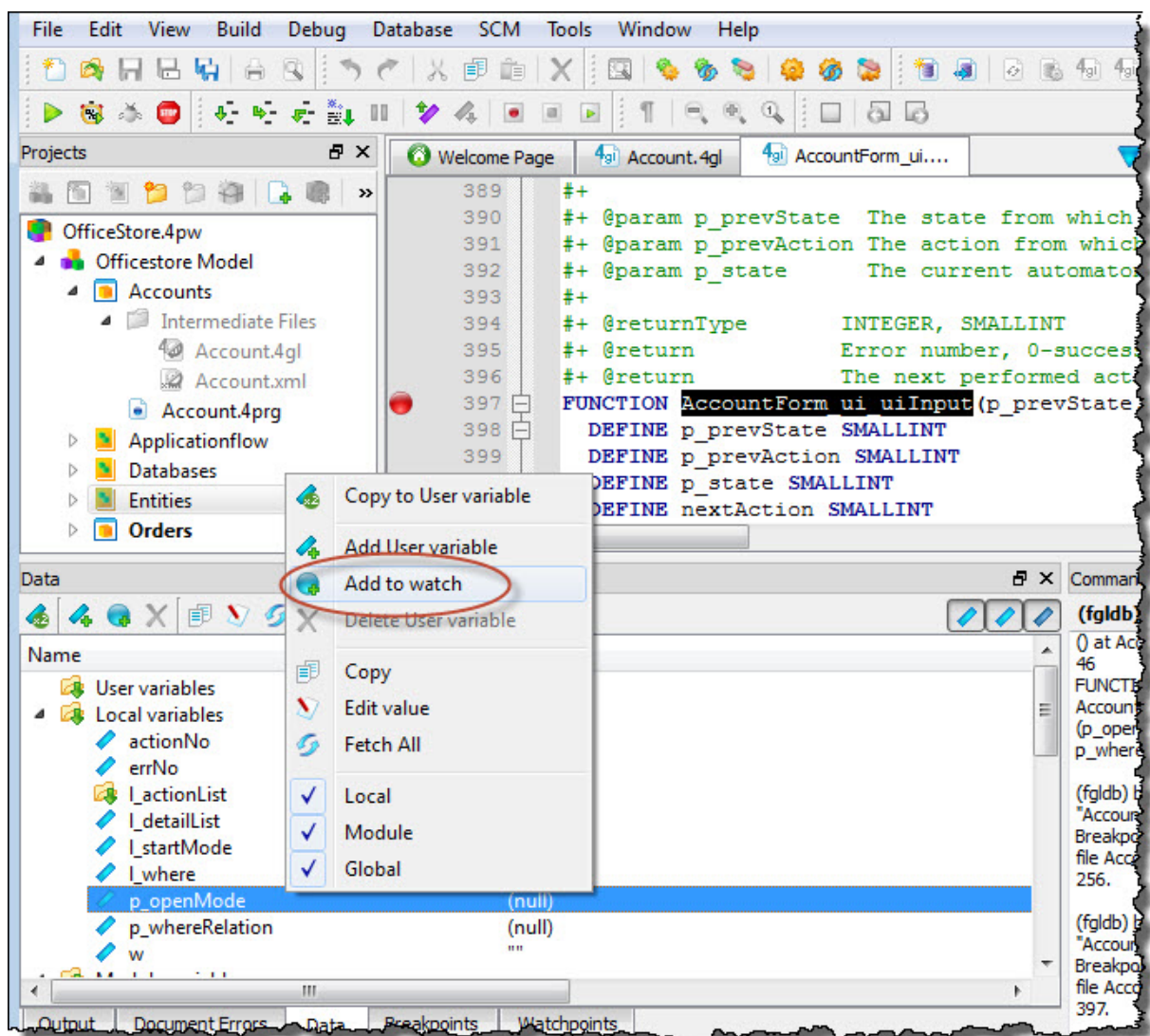


Figure 33: Adding a Watchpoint

8. In the command line area of the Command view, type `continue` to resume program execution until the `p_openMode` watchpoint is triggered.

Entering the `continue` command resumes program execution until a breakpoint is reached, a watchpoint is triggered, or the program terminates. The **continue** command is also available in the Debug menu and Toolbar.

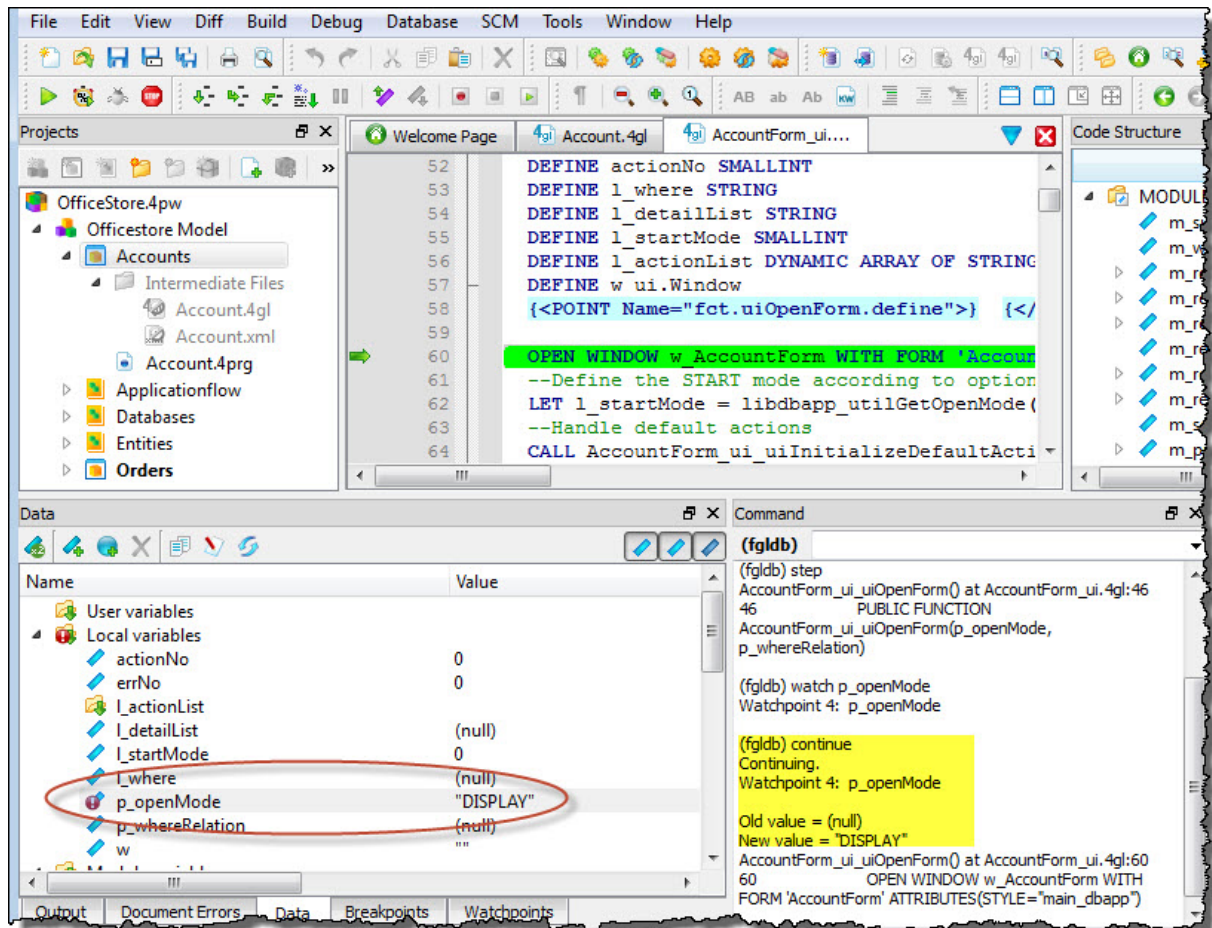


Figure 34: Command view showing the results of the `continue` command

9. Select `Debug >> Continue`.

An application screen displays and program execution stops at the breakpoint on function `AccountForm_ui_uiDisplay()`.

10. Use the `Backtrace` view to see which functions in the program have been called.

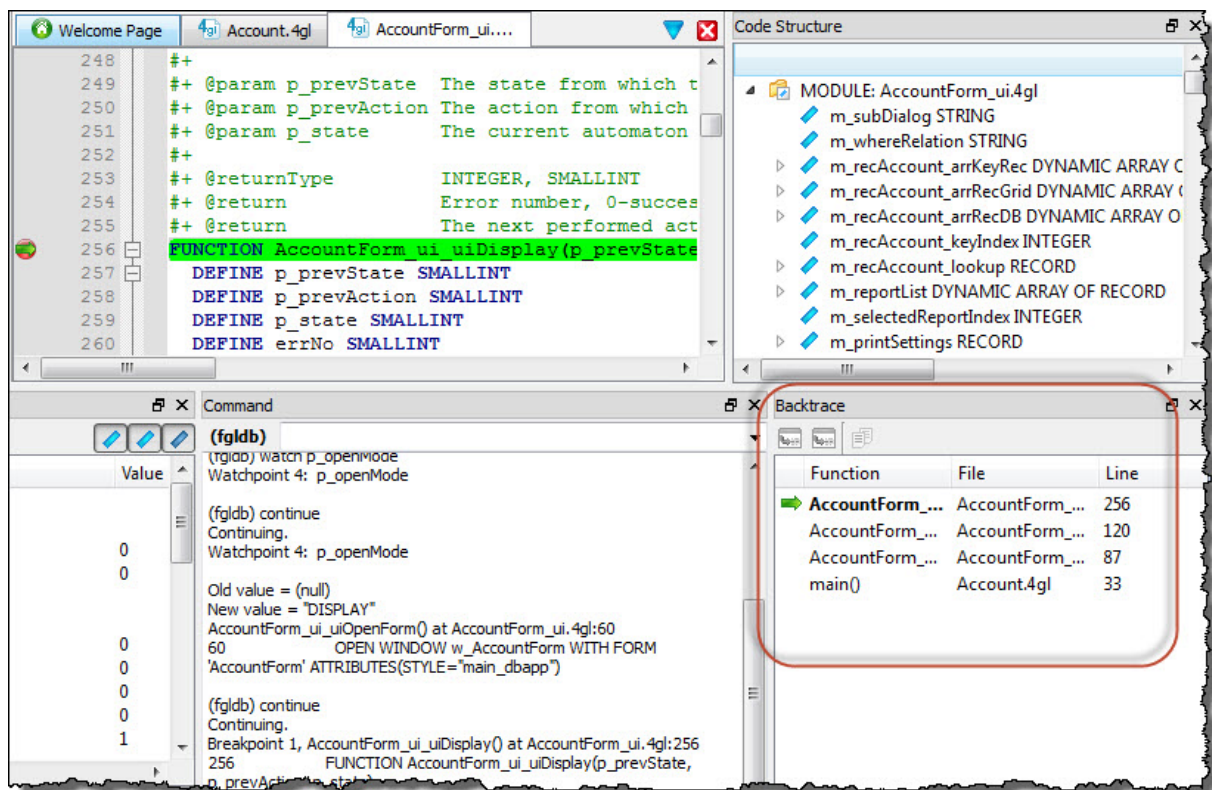


Figure 35: The Backtrace view

11. Select **Debug >> Continue**.

Program execution continues until an interactive Dialog statement (DISPLAY ARRAY) switches control to the application screen, which appears exactly as if it were running outside the Debugger.

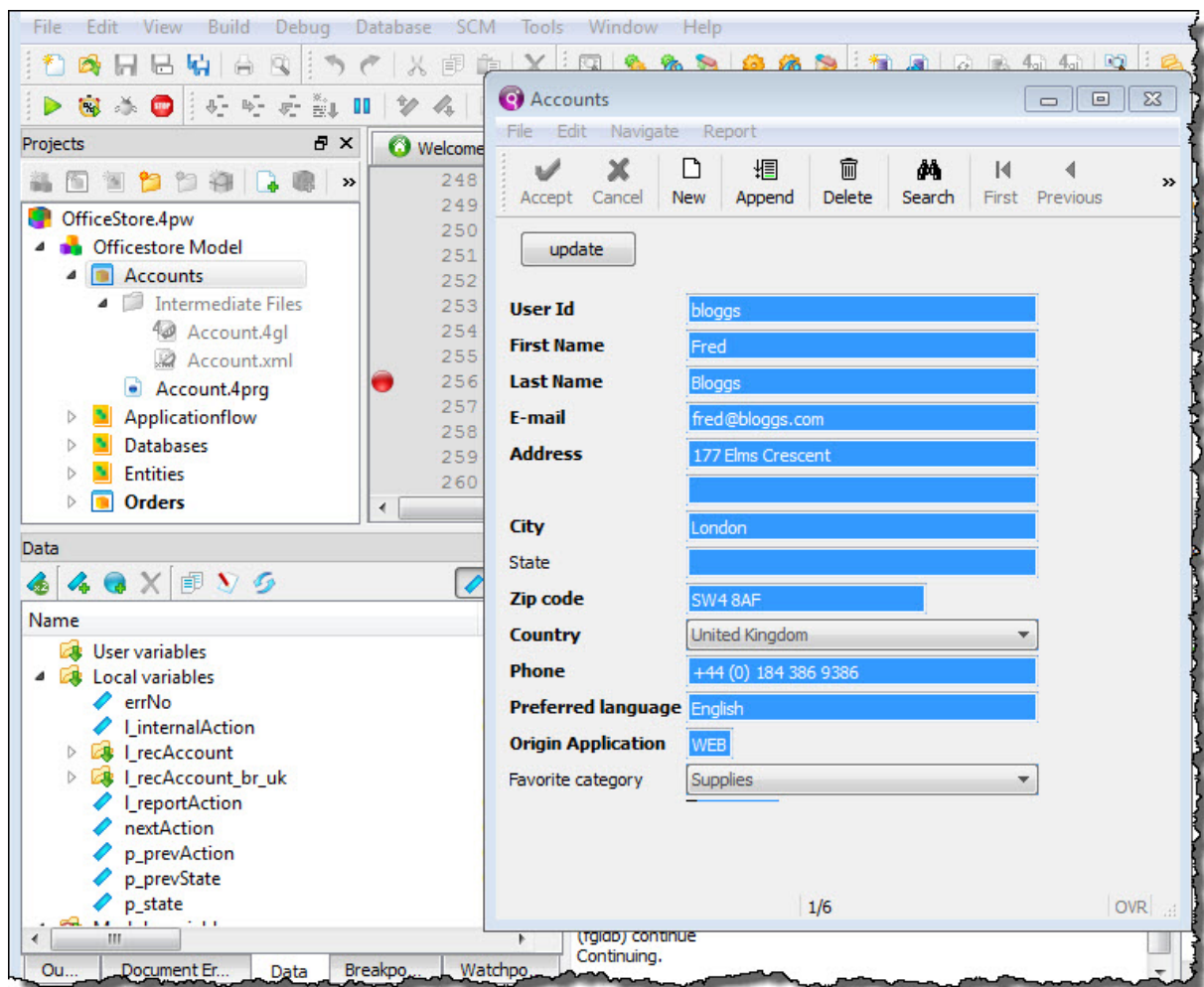


Figure 36: The application screen

12. Interact with the program:

- a) Select **Search** in the Accounts window
- b) Type **miller** into the User Id field and select **Accept**.

Control switches back to the Debugger and program execution continues until it reaches the breakpoint at function `AccountForm_ui_uiDisplay()`.

- c) Select **Debug >> Next** to step through the application one instruction at a time until the record for userid miller displays and control is switched back to the application screen.
- d) Select **File >> Exit** to close the application screen and return control back to the Debugger.

13. Select Debug >> Continue to terminate the application and complete the tour.

Terminating the application also terminates the Debugger session.

vi emulator and diff tools in Code Editor

Easily switch to the vi code editor while working with text files.

1. From the **Officestore** project, expand the **Orders** node and **Intermediate Files** folder. Double-click on `Order.4gl` to open it in Code Editor.
2. Add a new comment anywhere in the file by typing `--My comment`.
3. Select **Edit >> VI Editing Mode**. Use vi commands to navigate the file and remove the comment you added in the prior step.
4. Notice that the file is automatically in **diff mode**.

Explore Source Code Management

Maintain and share project files with Genero Source Code Management (SCM)

Before you begin, a Subversion client must be installed on your local machine as described in [What is Genero Source Code Management?](#) on page 529. In addition, you must have access to a Subversion repository containing the OfficeStore sample to perform the steps below.

The steps in this tour assume that the files for the OfficeStore sample project have been stored in an SVN Repository.

1. Use the **Checkout** option from the SCM menu to checkout the OfficeStore project files from the repository to your checkout directory.

The SCM Checkout wizard steps you through the process of checking out project files and prompts you to select a project file (4pw) to open in Project Manager.

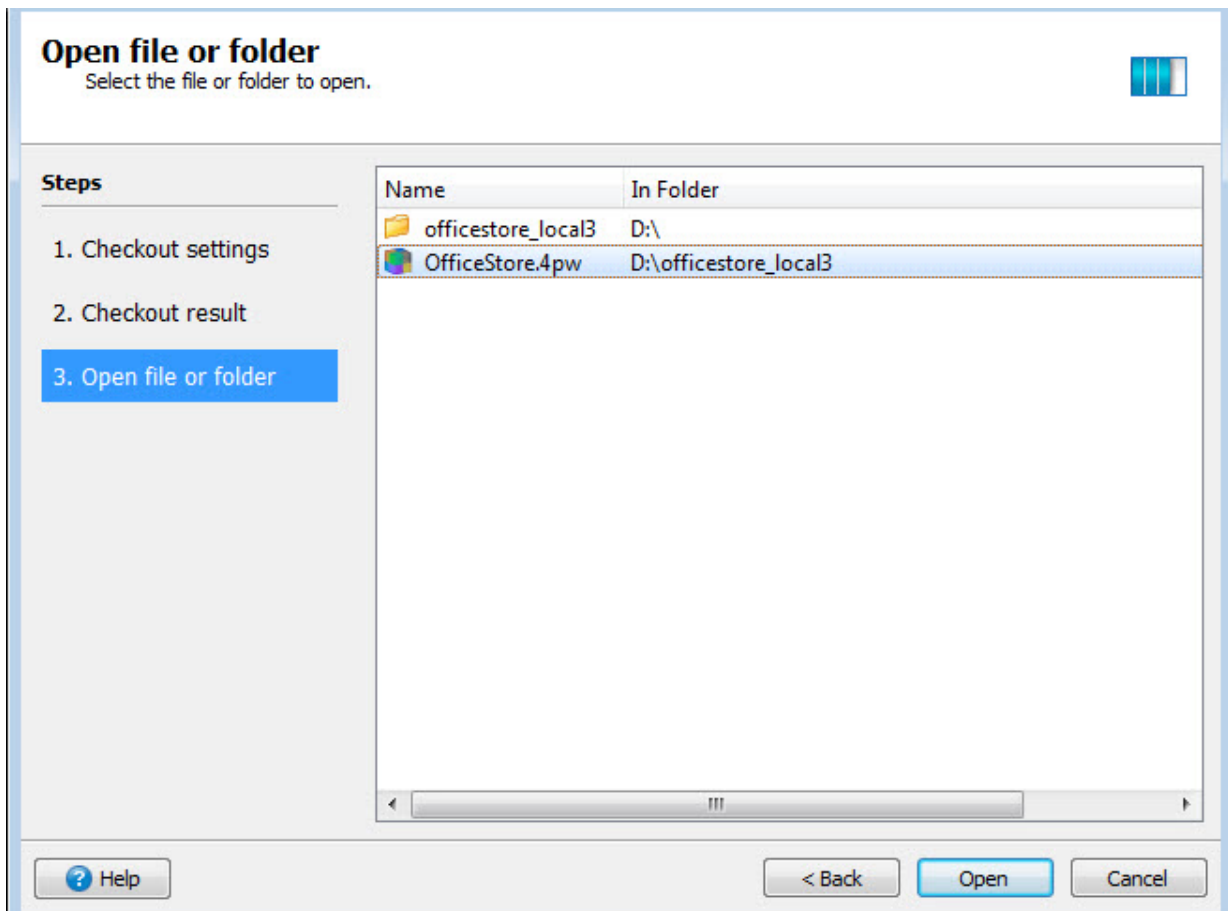


Figure 37: The SCM Checkout wizard

2. Select **File >> New >> Genero Files, Style (.4st)**, and save the new styles file as `account.4st` in the Entities node.

Adding a file to the project automatically performs an SVN `add`, which uploads and adds the file to the repository on your next SVN `commit`. The Entities folder displays a red exclamation mark to indicate uncommitted changes to the directory.

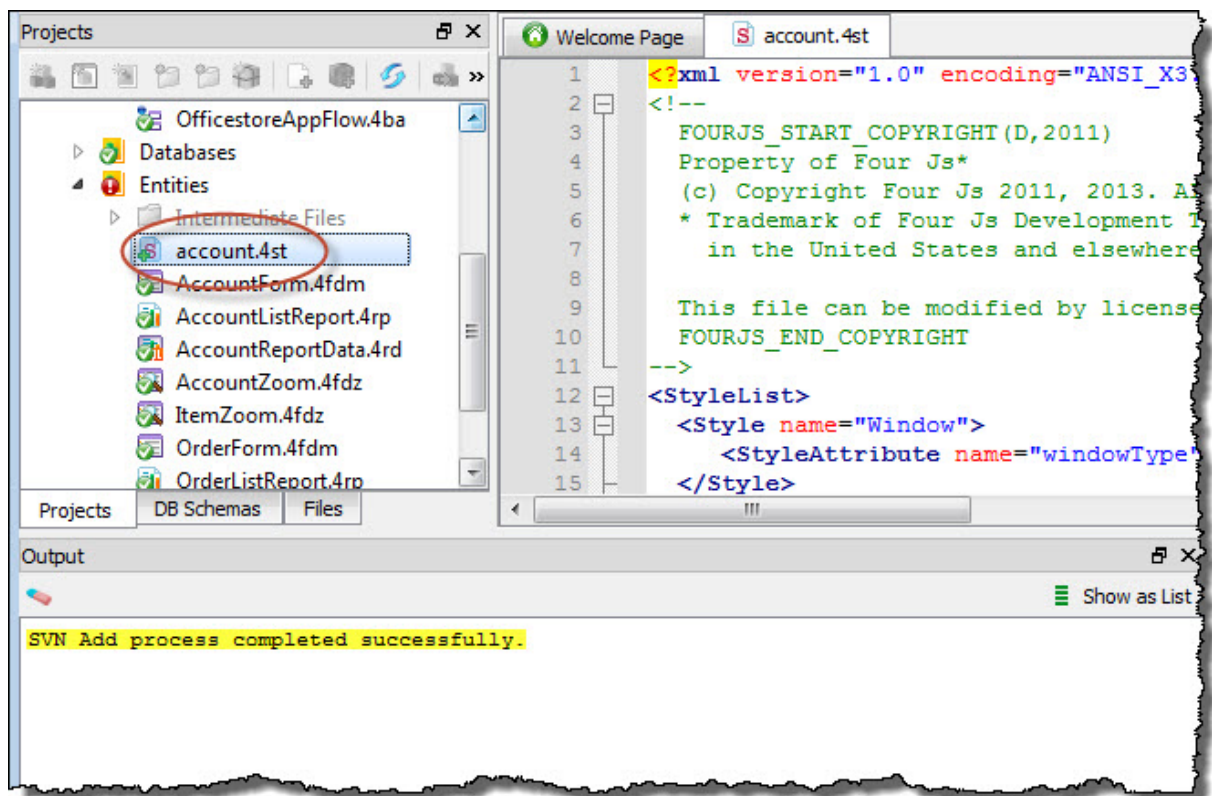


Figure 38: Adding a file to a project managed by SCM

3. Select **Window >> Views >> SVN Status**, navigate to your checkout directory and select **Add mode**. Selecting Add mode lists the newly added styles file as well as any other unversioned files present in the directory.

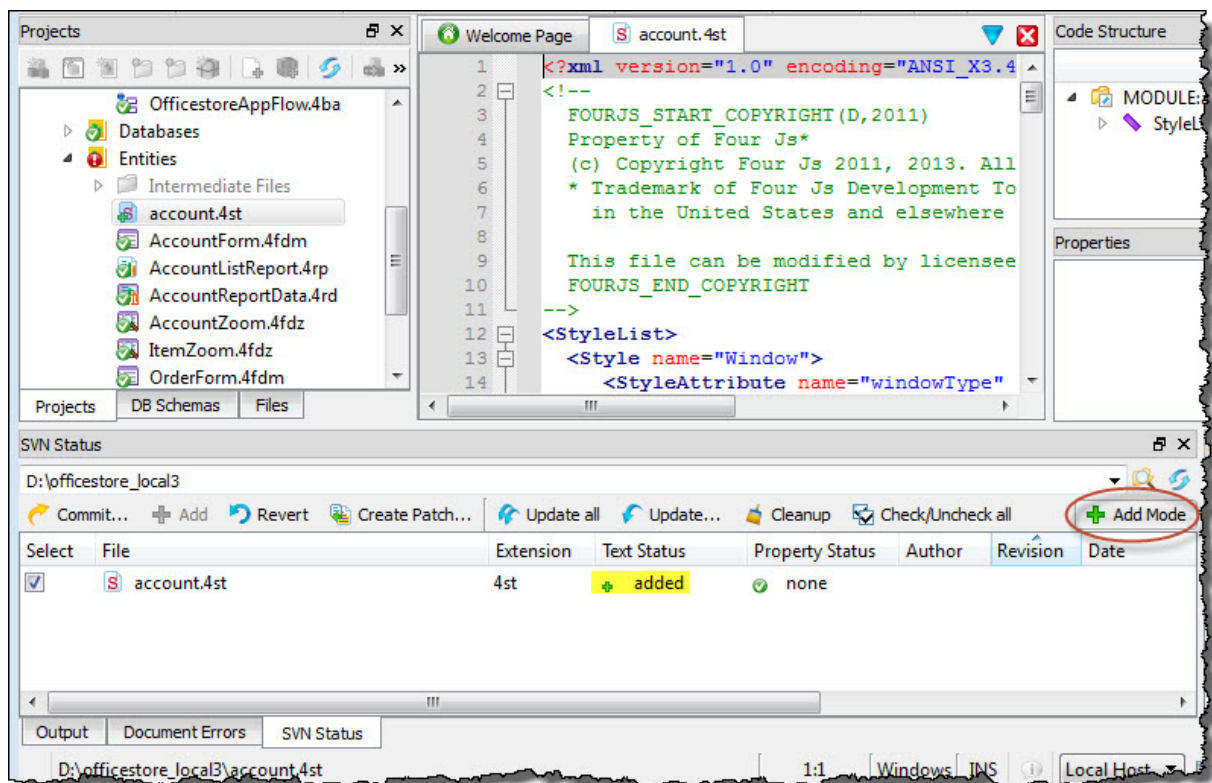


Figure 39: The SVN Status view

4. Ensure the **Select** option is checked and select the **Commit...** button at the top of the view to invoke the SVN Commit dialog.
5. Enter `New styles file` in the Commit Comments area and select the **Commit** button to commit the new file to the repository.
6. Enable the SVN needs-lock property on the `account.4st` file to prevent commit conflicts. Setting the needs-lock property requires a user to lock the file before modifying it.
 - a) Right-click on the `account.4st` file in the Projects view and select **SCM >> Properties**.
 - b) Select the **+** at the top of the Property view to launch the Add SVN Property dialog.
 - c) Select **svn:needs-lock** from the selection list, select **OK** and close the SVN Properties dialog.

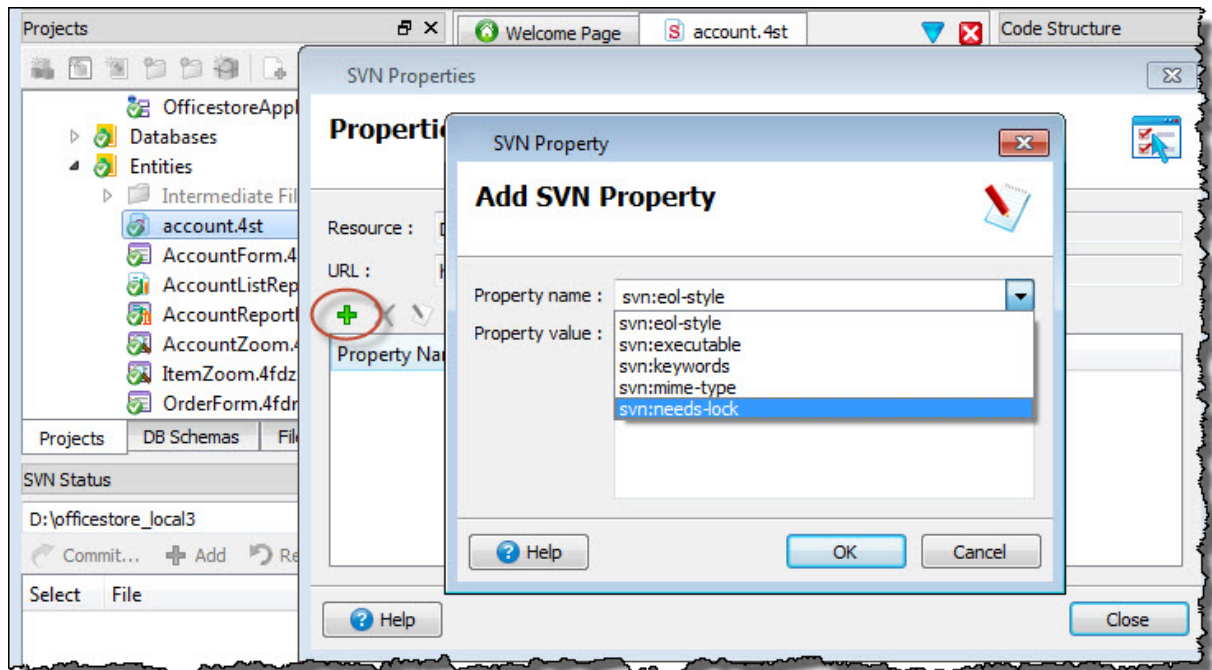


Figure 40: Setting the needs-lock property

7. Lock and edit the `account.4st` file.

- a) In Project Manager, right-click on the `account.4st` file and select **SCM >> lock**.
- b) Enter `testing locks` in the Lock files comment area and select **Lock**.
- c) Select the **SVN Locks** button in the SVN view at the bottom of the screen to view information about all locked files in the checkout directory.
- d) Open the `account.4st` file in Code Editor, delete lines two through nine and save the changes. Deleting the lines will create many errors as indicated by the red icons in the Code Editor gutter.

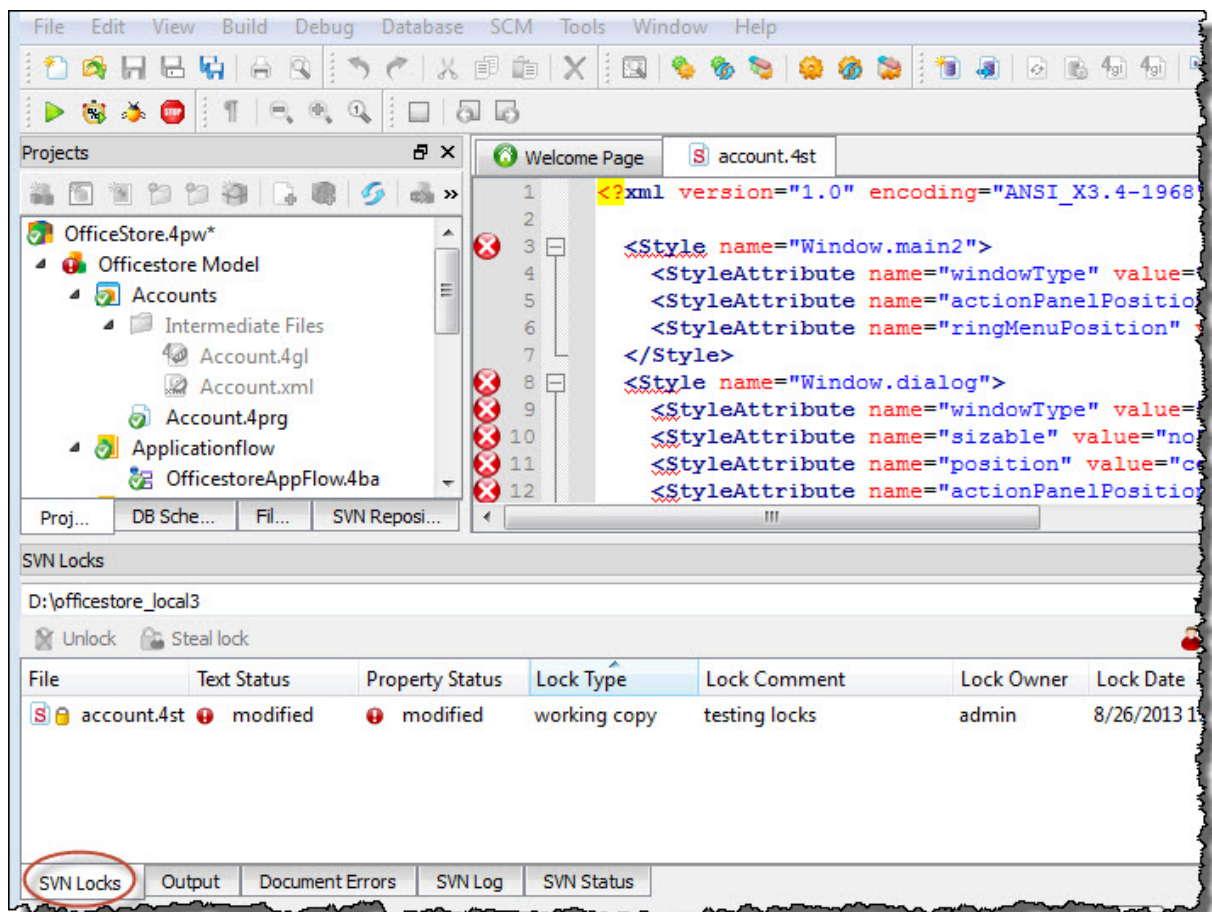


Figure 41: The SVN Locks view

8. Select the `account.4st` file in the SVN Status view and commit the changes (use `style` edits for the Commit Comment).

Note that committing the file releases the lock. You may be prompted to reload the modified file when you commit. Respond by selecting **Reload**.

9. Right-click the `account.4st` file in the Project view and select **SCM >> Show Log** to add and open the SVN Log view at the bottom of the screen.
10. Right-click on the entry for Revision 4 in the SVN Log and select **Diff** to call the Diff utility to compare the selected revision with the previous revision.

You can also use `ctrl-click` to select specific revisions you wish to compare with the Diff utility.

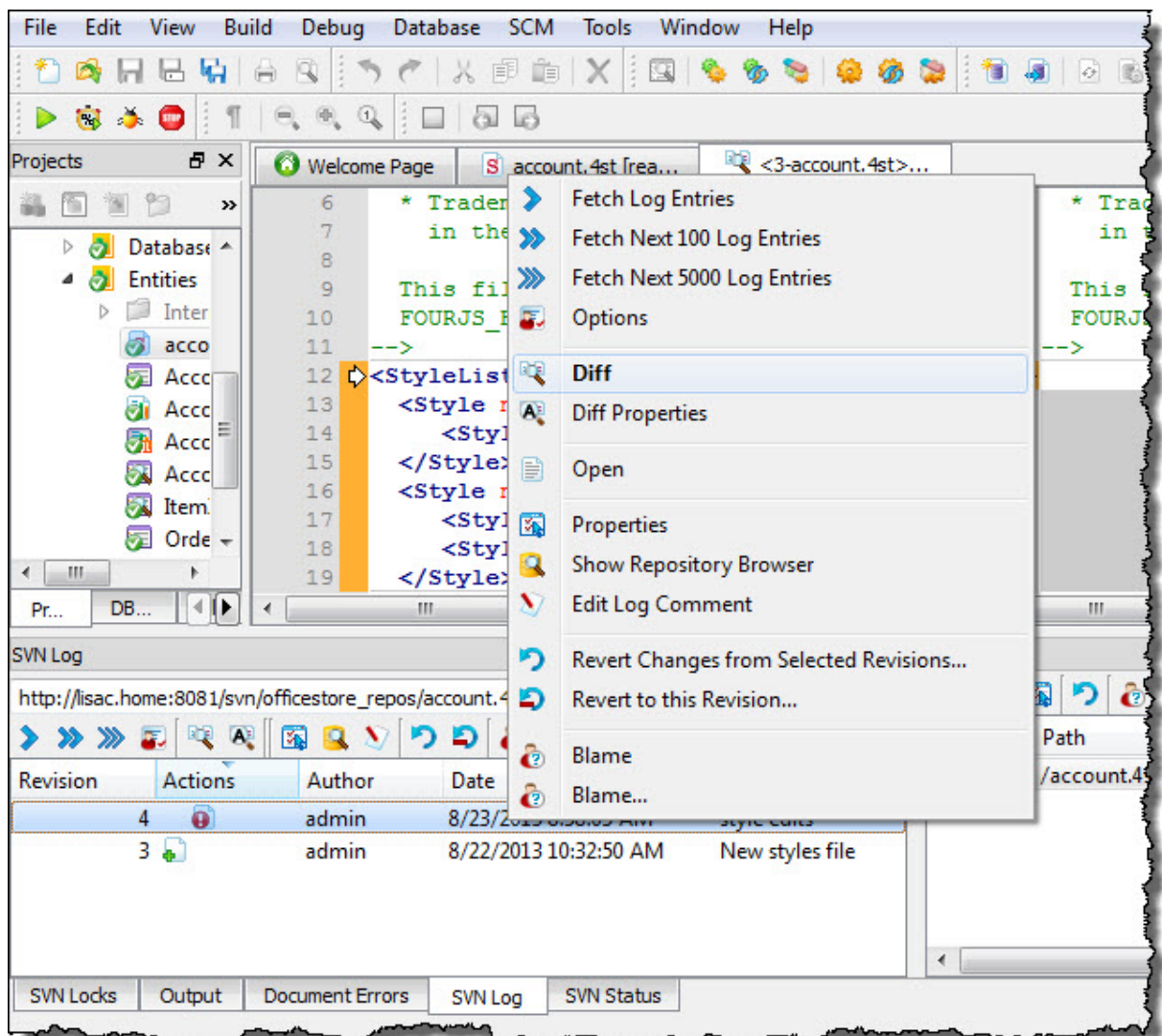


Figure 42: Performing a Diff

11. Right-click on the entry for Revision 4 in the SVN Log view and select **Blame** to see inline annotations for each change, including the author and revision number.

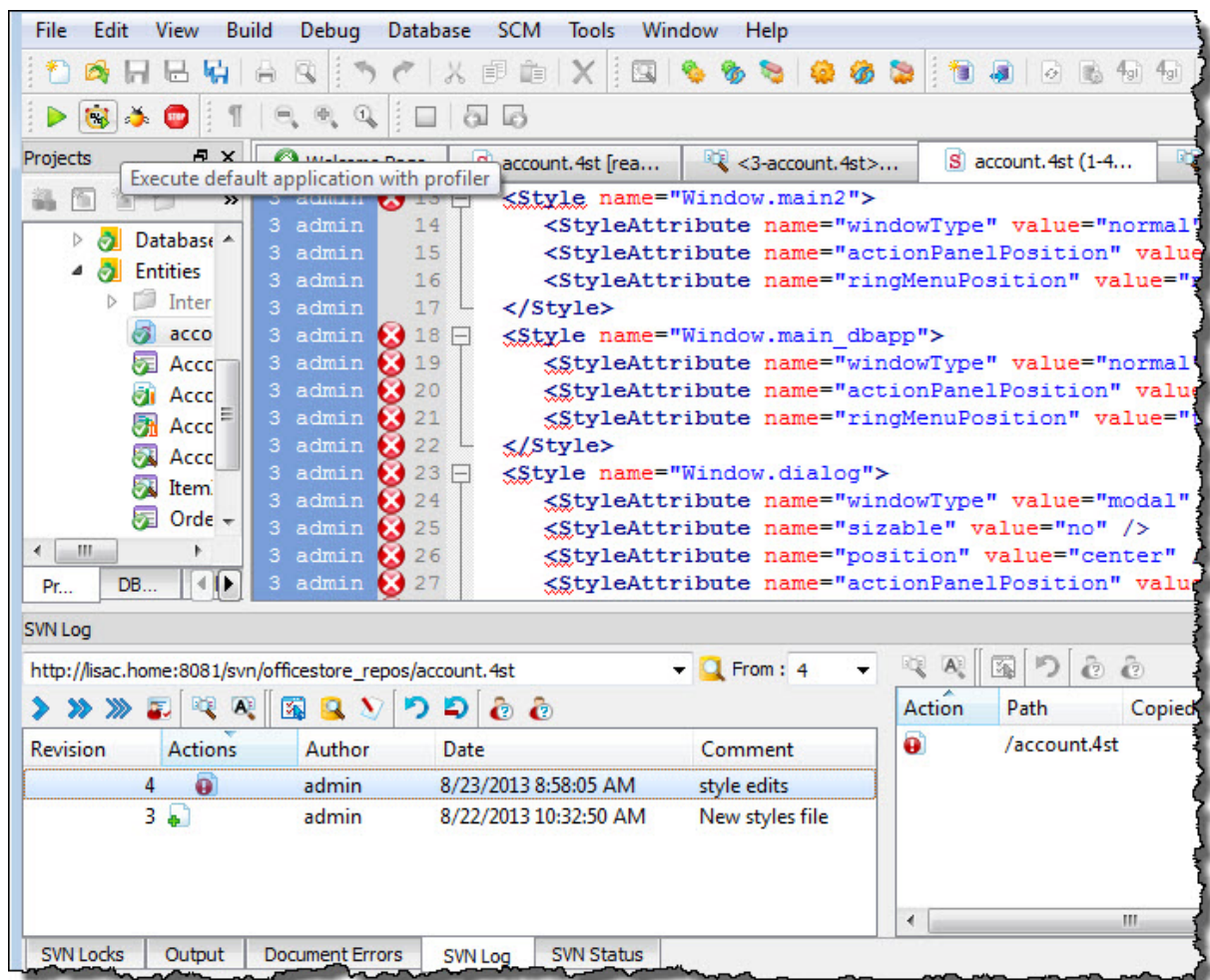


Figure 43: The Blame view

12. Revert back to an error-free version of the `account.4st` file.

- a) Right-click on the entry for Revision 3 and select **Revert to this Revision...** to launch the SVN Merge/Revert dialog.
- b) Accept the defaults in the SVN Merge/Revert dialog and select **Merge/Revert**, then **Finish** in the Merge/Revert result window.

The `account.4st` file reverts back to the selected revision and displays in Code Editor with the deleted lines from Revision 4 replaced and highlighted.

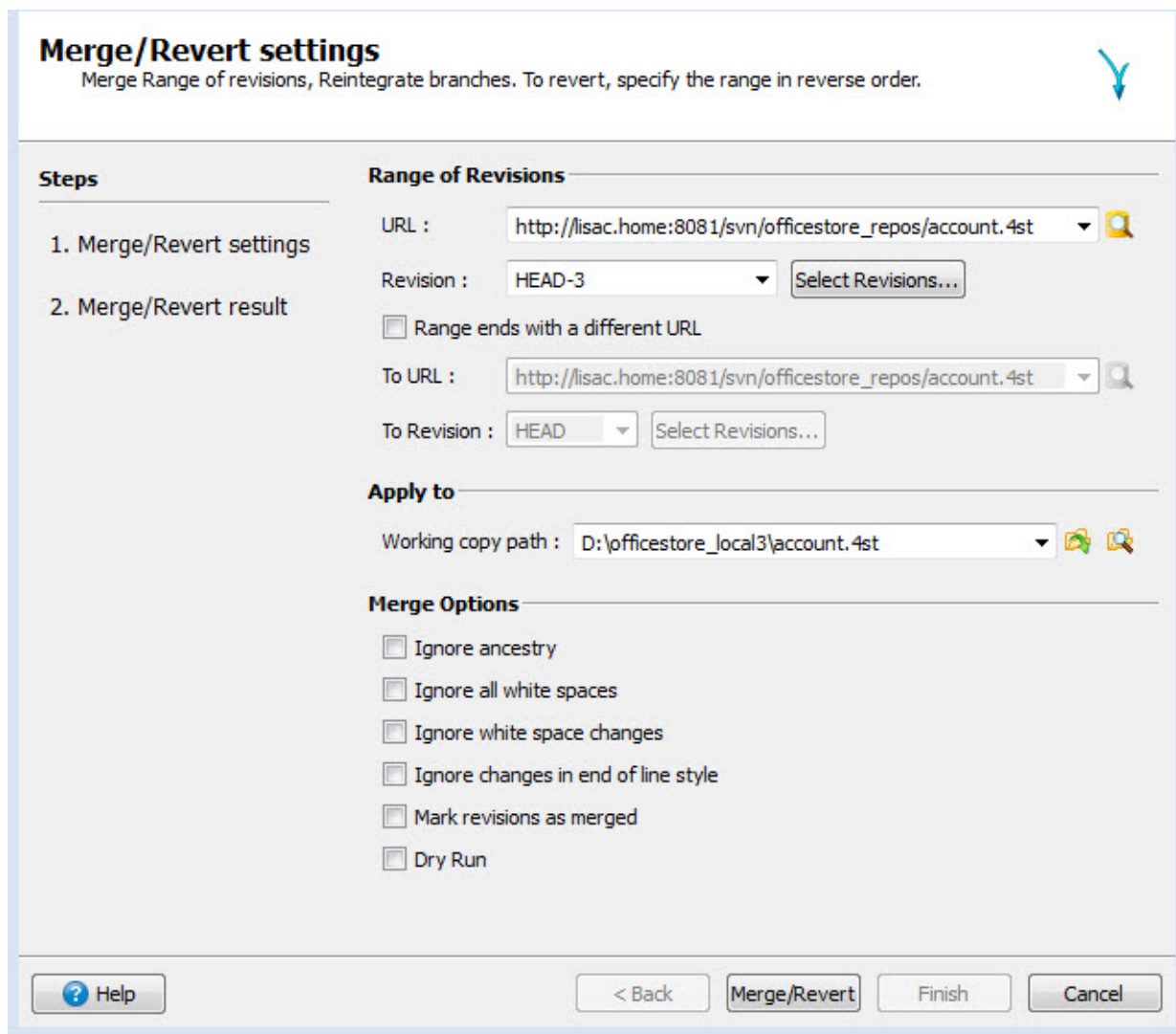


Figure 44: Reverting to a previous revision

13. Delete the `account.4st` file to complete the tour.
 - a) Close the `account.4st` file in the Code Editor view.
 - b) Right-click the `account.4st` file in Project Manager and select **Delete from Disk**.
 - c) Select the `account.4st` file in the SVN Status view and select **Commit...** (enter `SVN test` for the Commit Comment).

Explore database meta-schemas

Follow these steps to interact with a database meta-schema for developing forms, reports, and services.

1. From the **Welcome Page, Projects** tab, select the **OfficeStore** project. This opens the **OfficeStore** project file in the project view.
2. Expand the project and find the **Databases** node.
3. Double-click on the `officestore.4dbx` file. The file opens in the Meta-schema manager.
4. Note the layout of the tables and columns and the defined join relationships between them. Zoom in and out of the diagram by holding down the **Ctrl** key while scrolling with the mouse.
5. Select **File >> Print Preview**. Note that you can print any diagram in a variety of ways.
6. Find the `account` table. Select the `lastname` column and modify its `column length` from 80 to 120.

- Save your changes. Note that the diagram reflects that the schema is different from the database by flagging the modified column.
- [Generate a script](#) to update the database with your change to the meta-schema. Select **Database >> Generate Database Update Script**. Select OK and note that a .4gl file is created for you to use or expand upon as needed. Note the additional options in the Database menu allow you to also update a schema from the database as well as compare two schemas.

Explore forms

Follow these steps to explore the visual tools for user interface development.

Explore a form, then create a new form

- Expand the **OfficeStore** project and find the **Entities** node.
- Expand the **Entities** node and double-click on the `OrderForm.4fdm` file.

The `OrderForm.4fdm` file is a form and opens in the Form Designer.

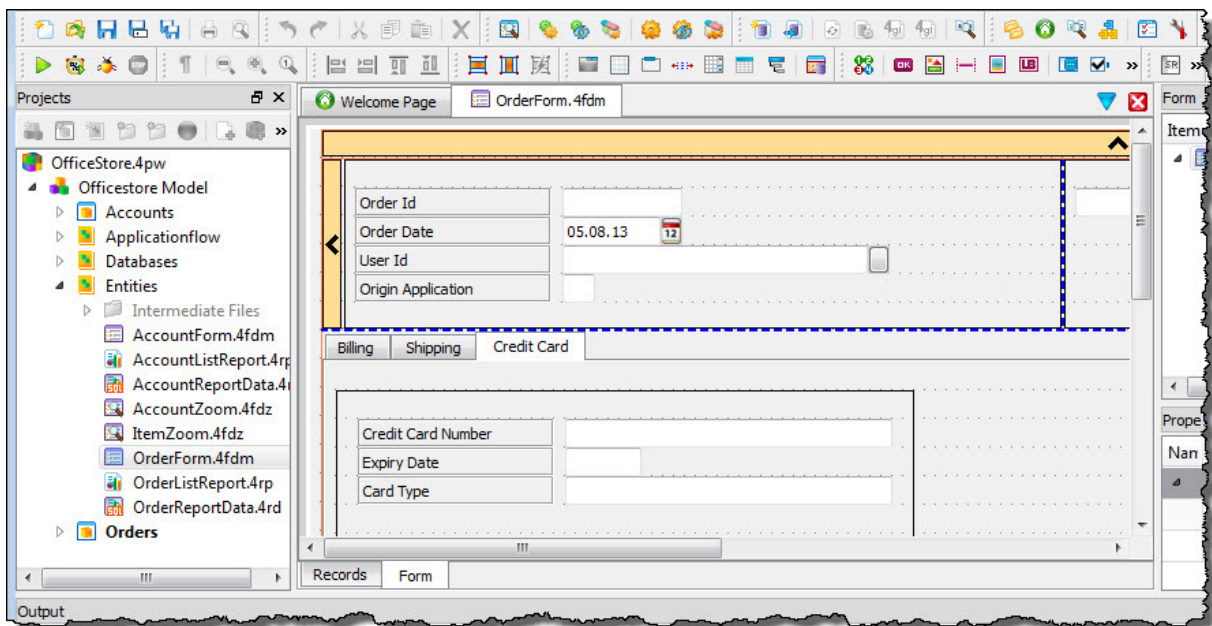


Figure 45: The OrderForm.4fdm file

- Customize your working environment to best suit you.
All panels can be re-sized with the mouse or closed and reopened with **Window >> Views**. Try pressing **Alt-F11** to toggle to Document editing. Select **Alt-F11** again to return to normal view. Double-click on the title of the Form Structure view to undock it. Double-click again to re-dock it into its last position.
- Scroll down in the Form Structure view to the Form section and select a form item.
The item is also selected in the form design.

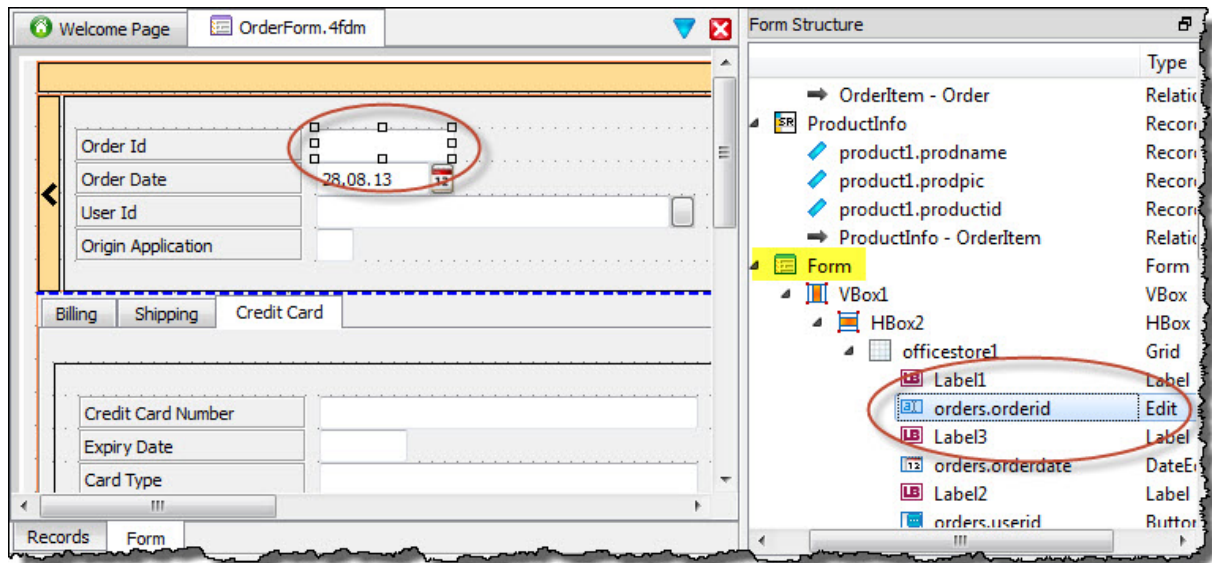


Figure 46: Selecting a form item in the Form Structure view

5. Use the **Form from Database** wizard to create a new form:
 - a) Select **File >> New >> Genero Files, Form from Database (.4fd)**.

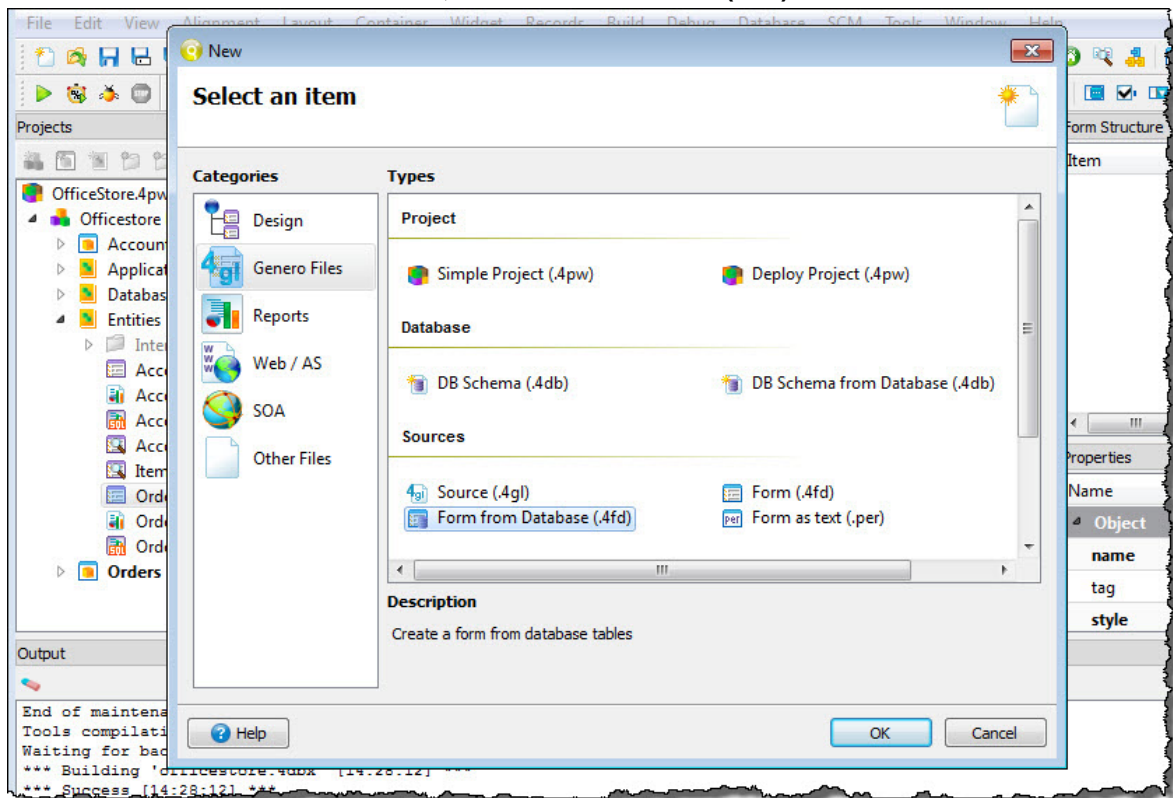


Figure 47: Creating a new form

- b) Select the **officestore** database and the **account** table (use the double arrow to select all columns in the account table for the form).

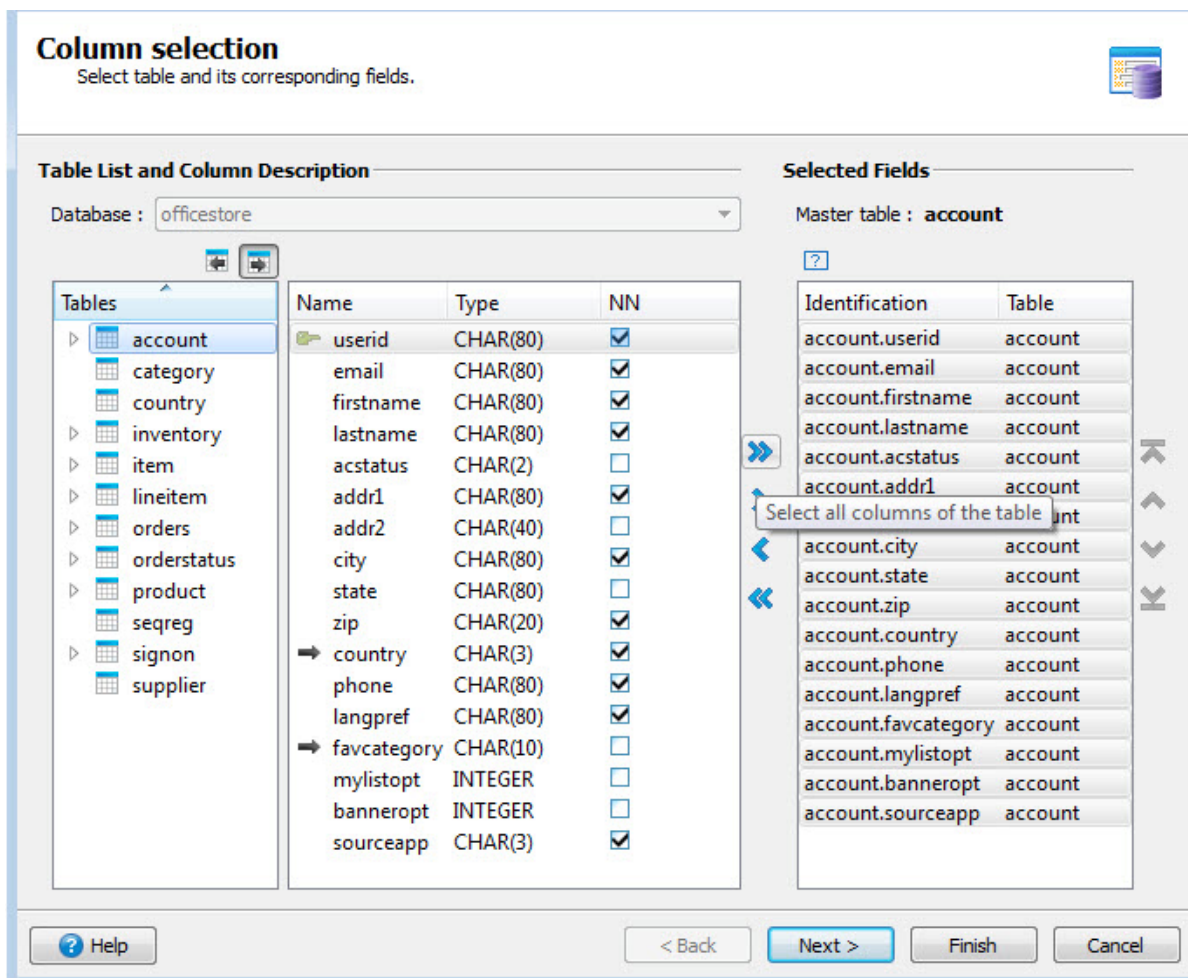


Figure 48: Selecting database and columns for a new form

c) Select Finish.

The form is now ready to use or modify in Form Designer. Additional table columns, container, and widgets can be added.

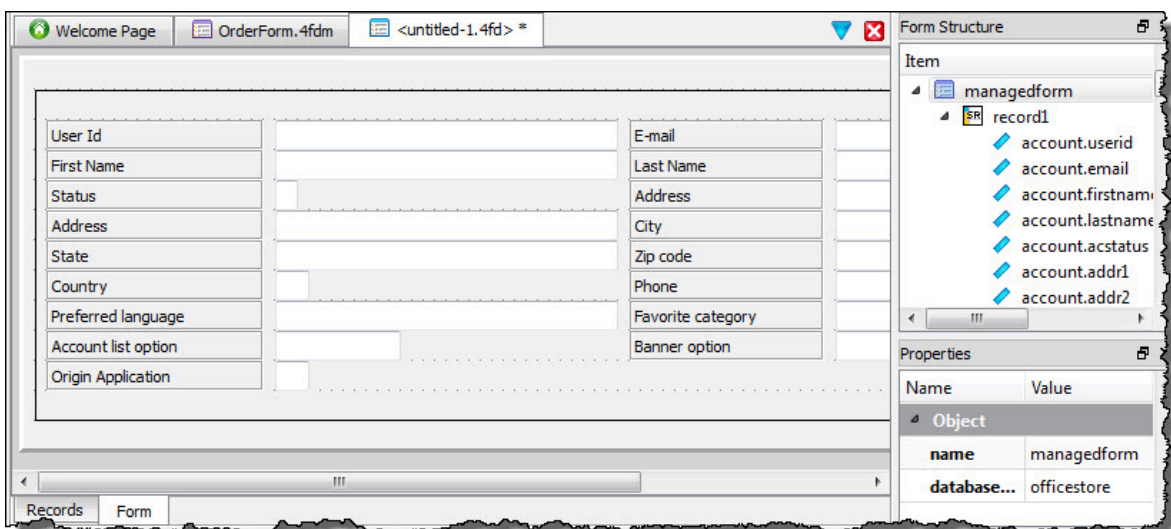


Figure 49: Form created from the Form from Database wizard

6. Select the Records tab at the bottom of the window.

The Records tab represents the data set for your form. Additional records can be added here or in the design.

7. Right-click on the record and select Add Field.
A field is added to the record with default values.

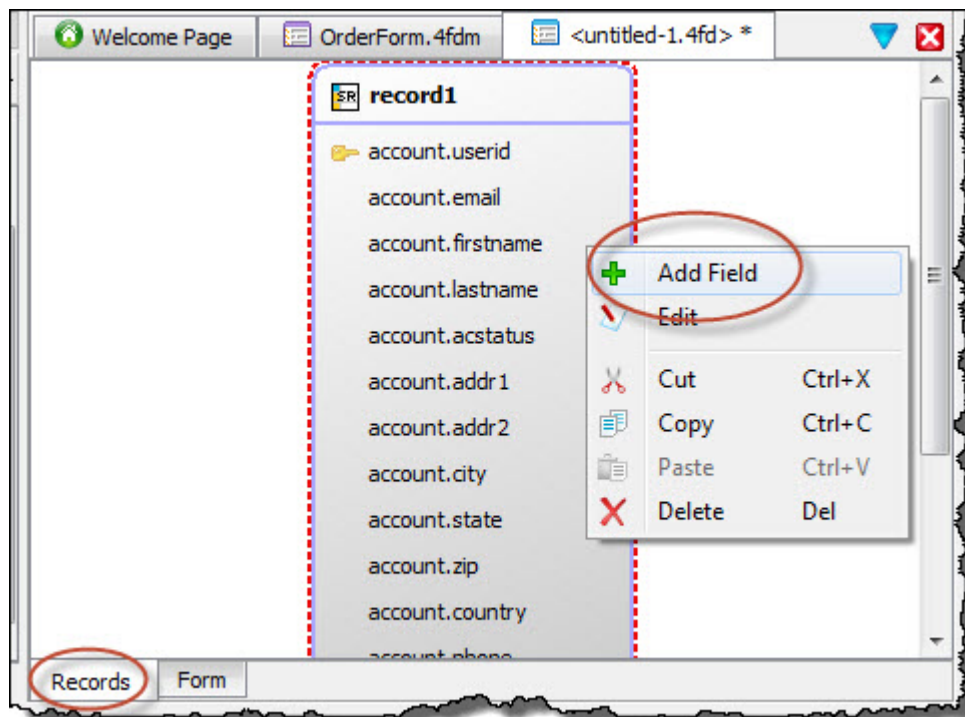


Figure 50: Adding a new field to a form record

8. Switch to the Form tab.
Note that the new field has been placed on the form in the top left. This field's properties and placement can be modified.
9. Right-click on the new field and select **Convert Widget >> FFLabel** to change the field to a form field label.

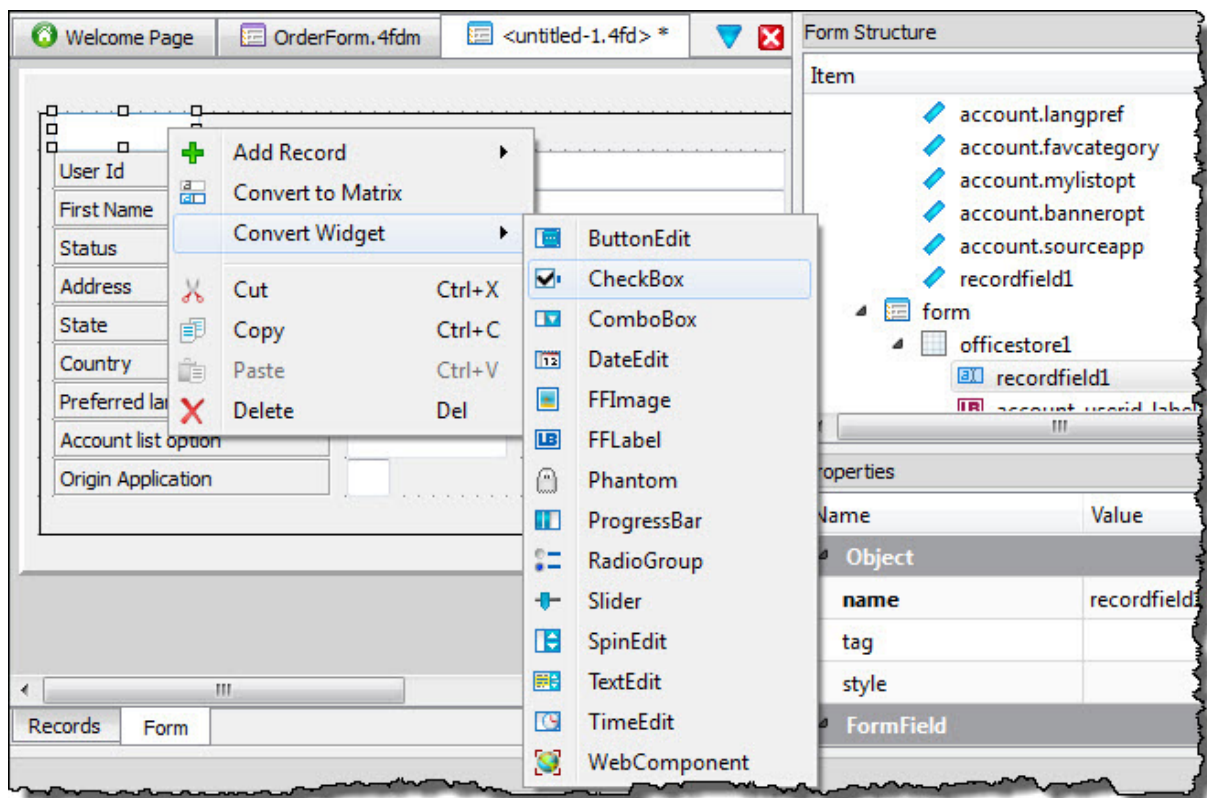


Figure 51: The Convert Widget menu

10. In the properties for the new field, change its color value from Black to Blue. The color changes on the form design.

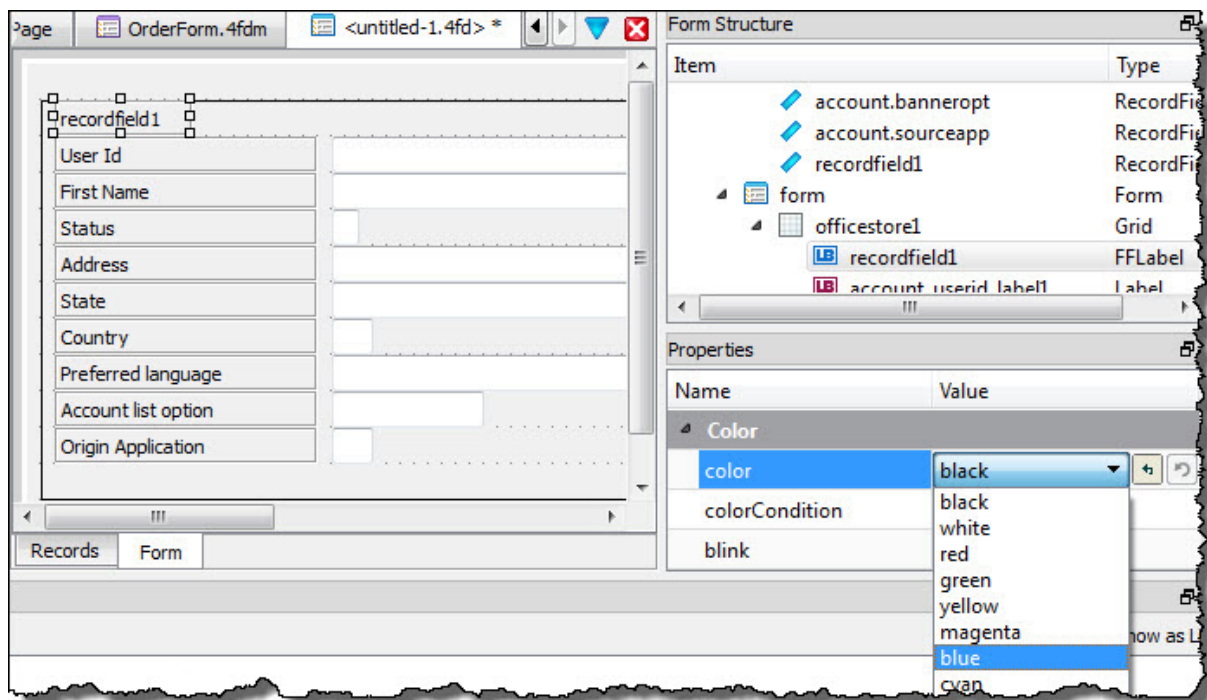


Figure 52: The color property

11. Select **Build >> Preview** to get an idea about how the form will render in the Desktop client.
12. Close the form preview and the untitled form (do not save it) to complete the tour.

Analyze code

Follow these steps to learn how to reverse engineer an application with Dependency and Sequence diagrams, resources provided by Genero Code Analyzer.

1. Right-click the **OfficeStore Group node** and select **Open Dependency Diagram**.

Dependency diagrams display a graphical view of the complex relationships between components of a project and can be opened at the group or application level.

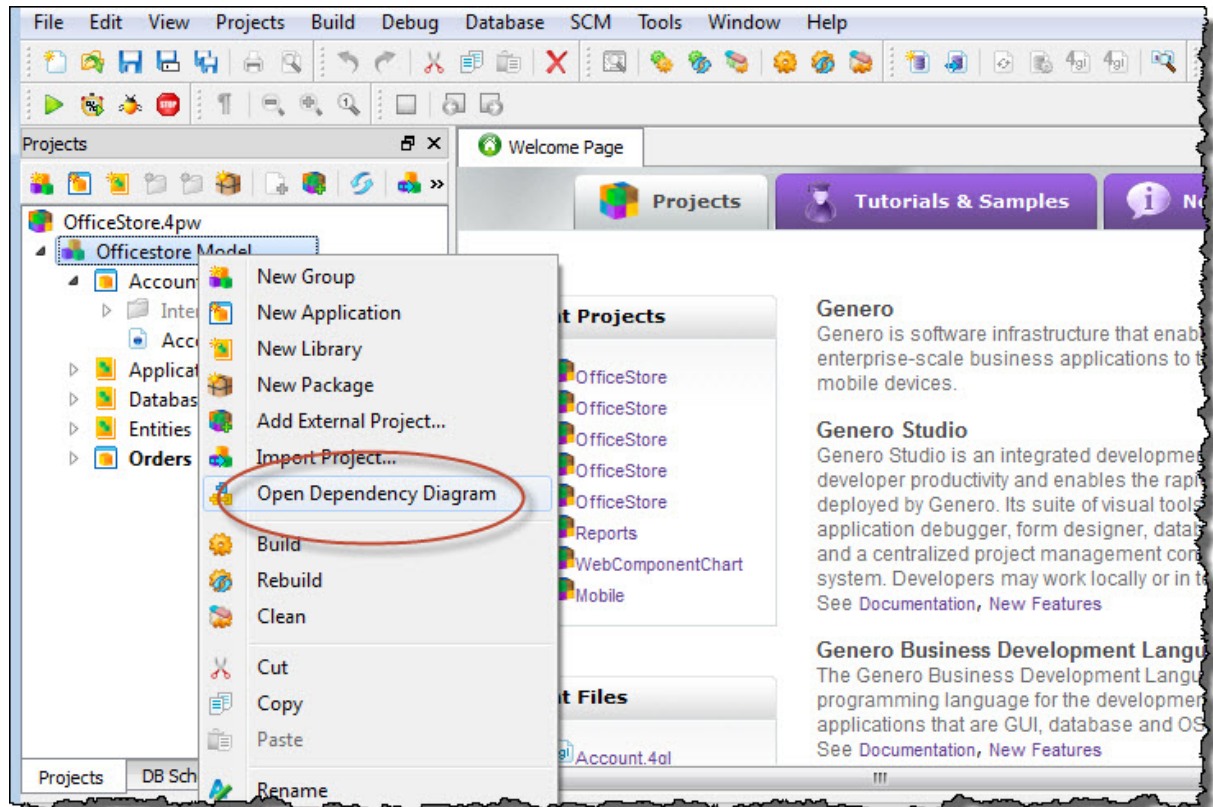


Figure 53: Option to open a Dependency Diagram

2. Right-click the **Entities node** in the Dependency diagram and select **Expand** from the context menu. The Entities node expands to display all the sub-components and the relationships between them. Use Ctrl-mouse wheel to zoom in and out.

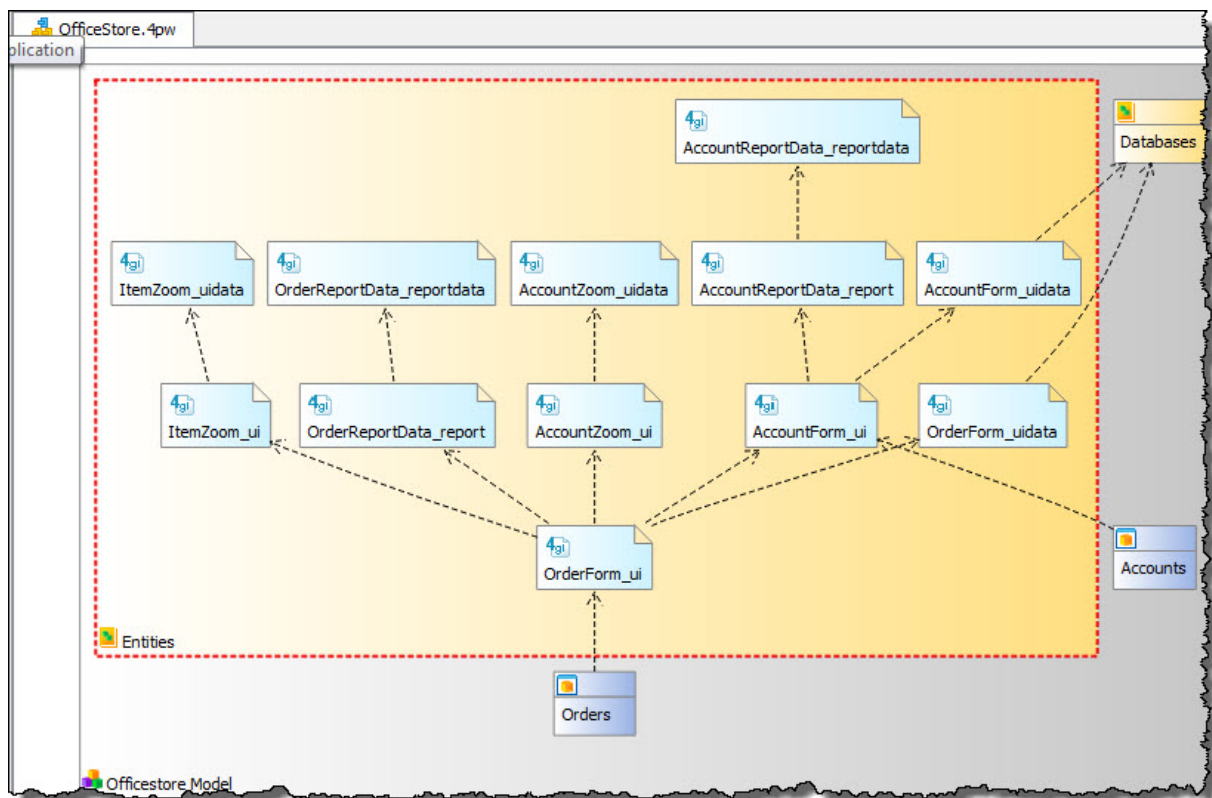


Figure 54: The expanded Entities node

3. Use a Dependency diagram filter to focus on the dependencies between components of the Accounts application:
 - a) Right-click anywhere in the Dependency diagram margins and select **Filter Items...**
 - b) In the **Select items to filter** dialog, deselect **Orders**.
 - c) Press **OK**.

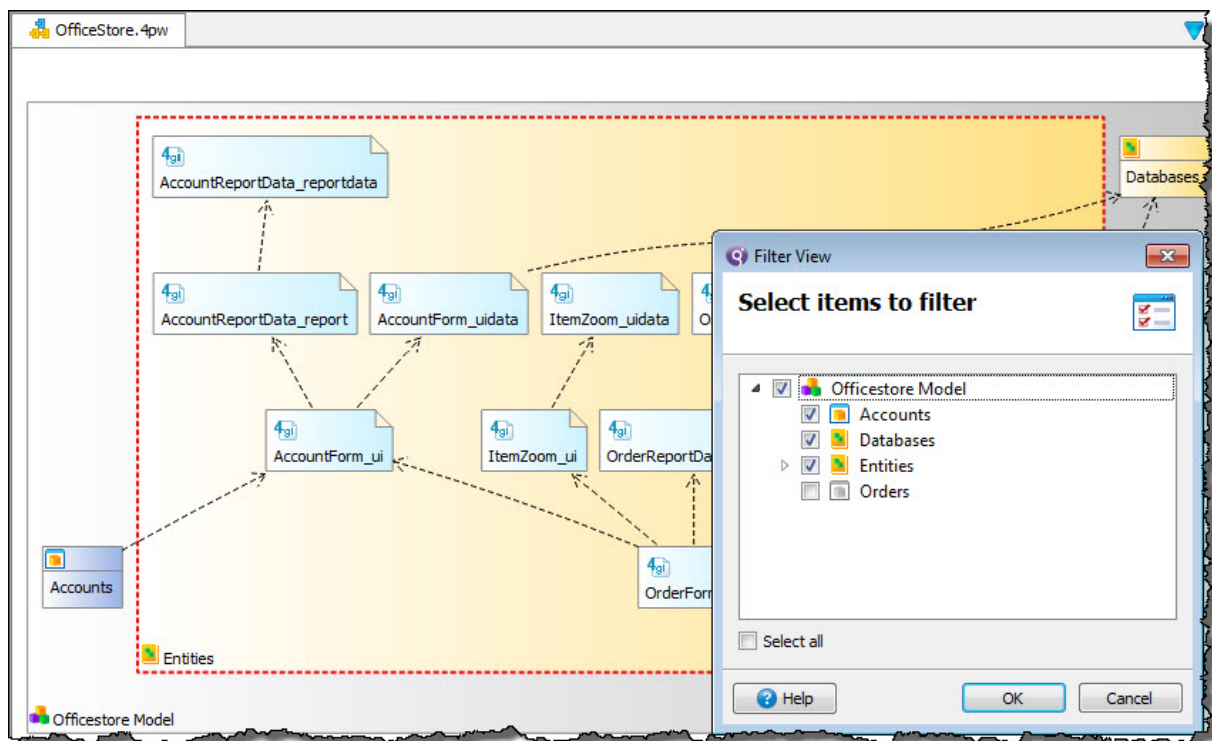


Figure 55: The Filter view

4. Select the link between `AccountForm_ui` and `AccountForm_uidata` to display associated function calls in the Function calls view.

Details about function calls between the selected modules are shown in the Function calls view and the project structure displays as a tree in the Dependency Diagram Structure view.

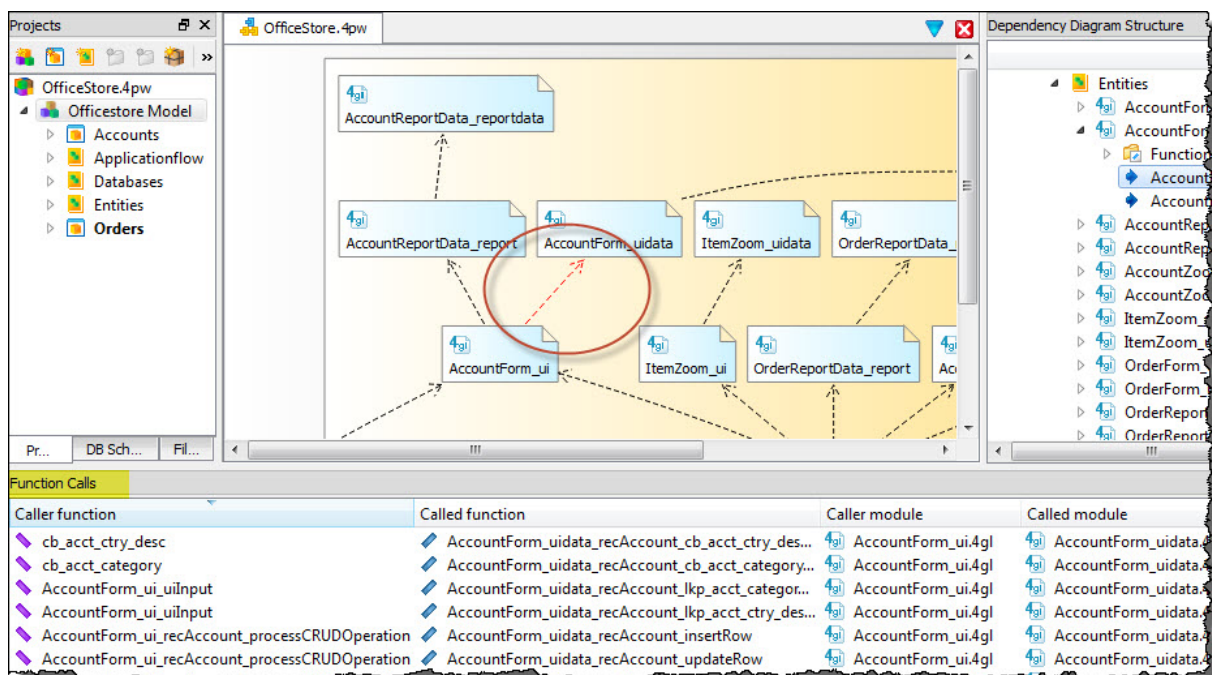


Figure 56: The Function Calls view

5. Right-click on the `AccountForm_ui_uiInput` function in the Function Calls or Dependency Diagram Structure view and select Open Sequence Diagram.

The diagram shows the logic flow of `AccountForm_ui_uiInput`, with the starting point indicated by the stick-figure representing the user who interacts with the application. The boxes represent functions in the `AccountForm_ui.4gl` module, and the sequence is indicated by the order in which the boxes are listed. Plus/minus signs on each box allow you to display or hide sub calls.

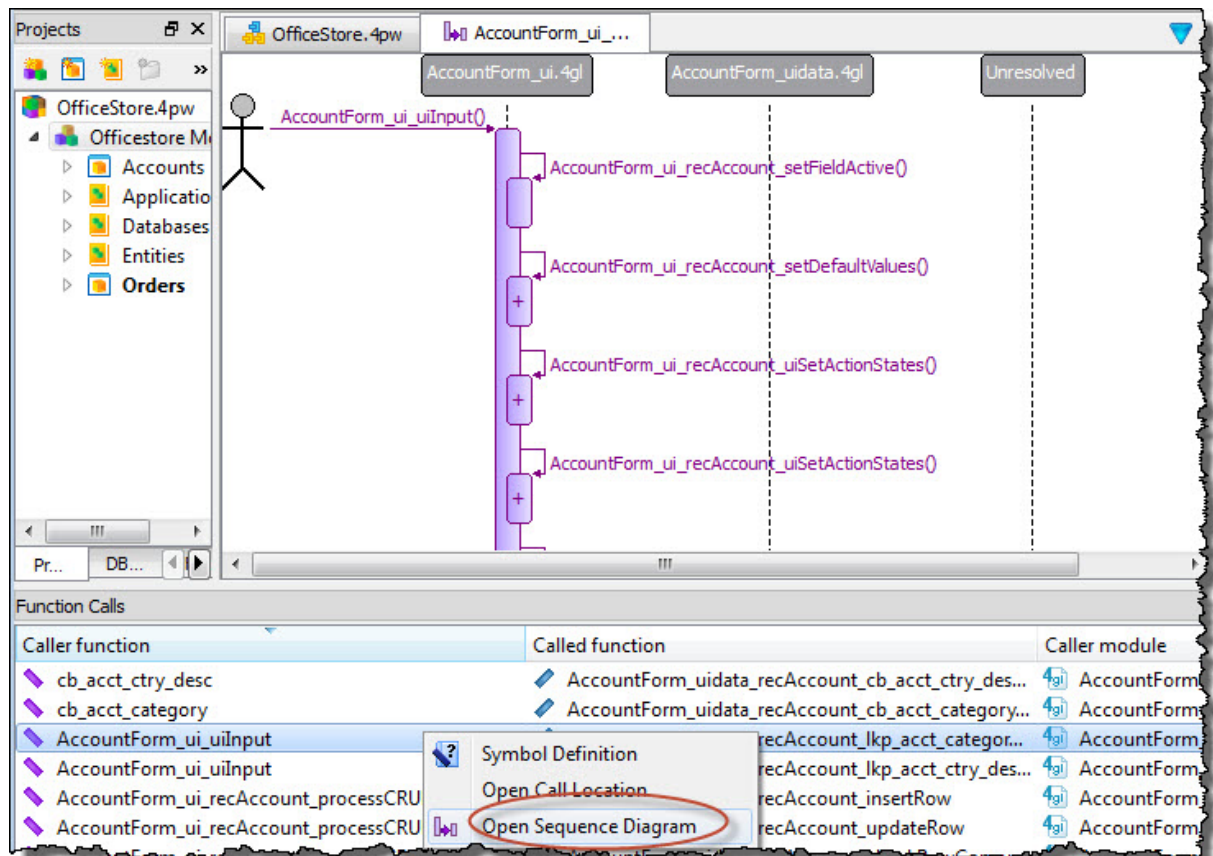


Figure 57: A Sequence diagram

- Right-click the box for the `AccountForm_ui_recAccount_setDefaultValues` function and select **Show Sub Calls**.

The box is expanded to show the `AccountForm_ui_recAccount_clearCascade()` subcall.

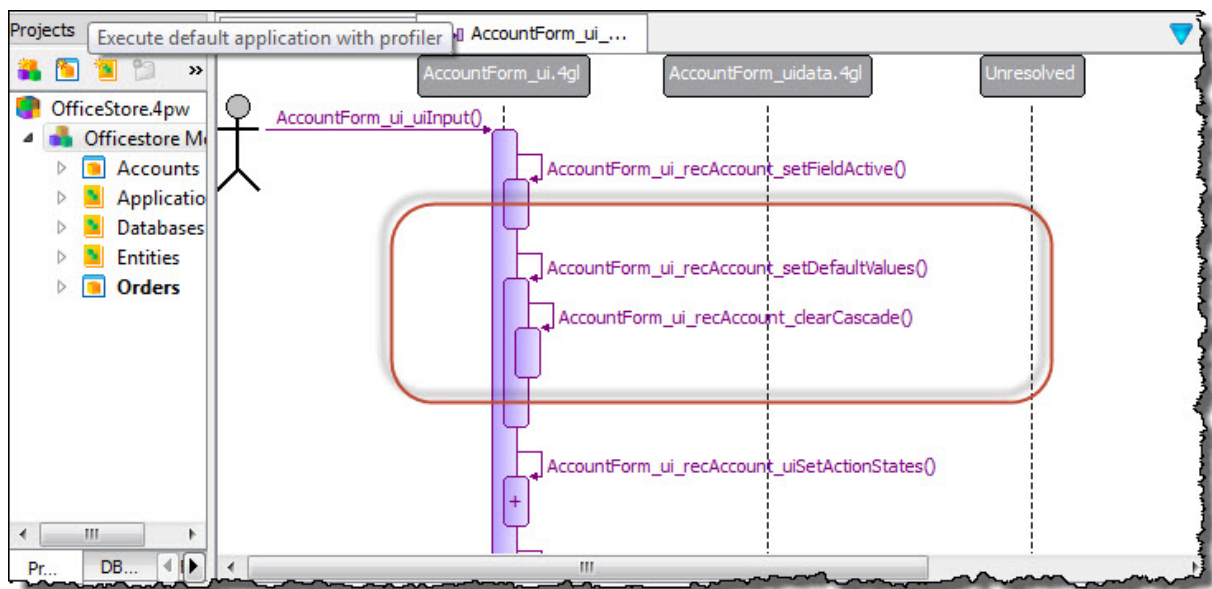


Figure 58: Sequence diagram with expanded function box

7. Close the `AccountForm_ui_uiInput` Sequence diagram tab and `OfficeStore.4pw` Dependence diagram tab in the Code Editor view to complete the tour.

Generate code

Genero Studio includes the Business Application Modeler (BAM) for generating business applications.

See [Quick Start: Generate an application](#) on page 176 or [Quick Start: Generating a mobile app](#) on page 182.

The Genero Studio framework

When a Genero Studio module is launched, the framework is displayed and all other windows and views, menus, Toolbars, and icons are contained within.

- [The Welcome Page](#) on page 70
- [Modules](#) on page 71
- [Toolbars and Menus](#) on page 82
- [Views](#) on page 96

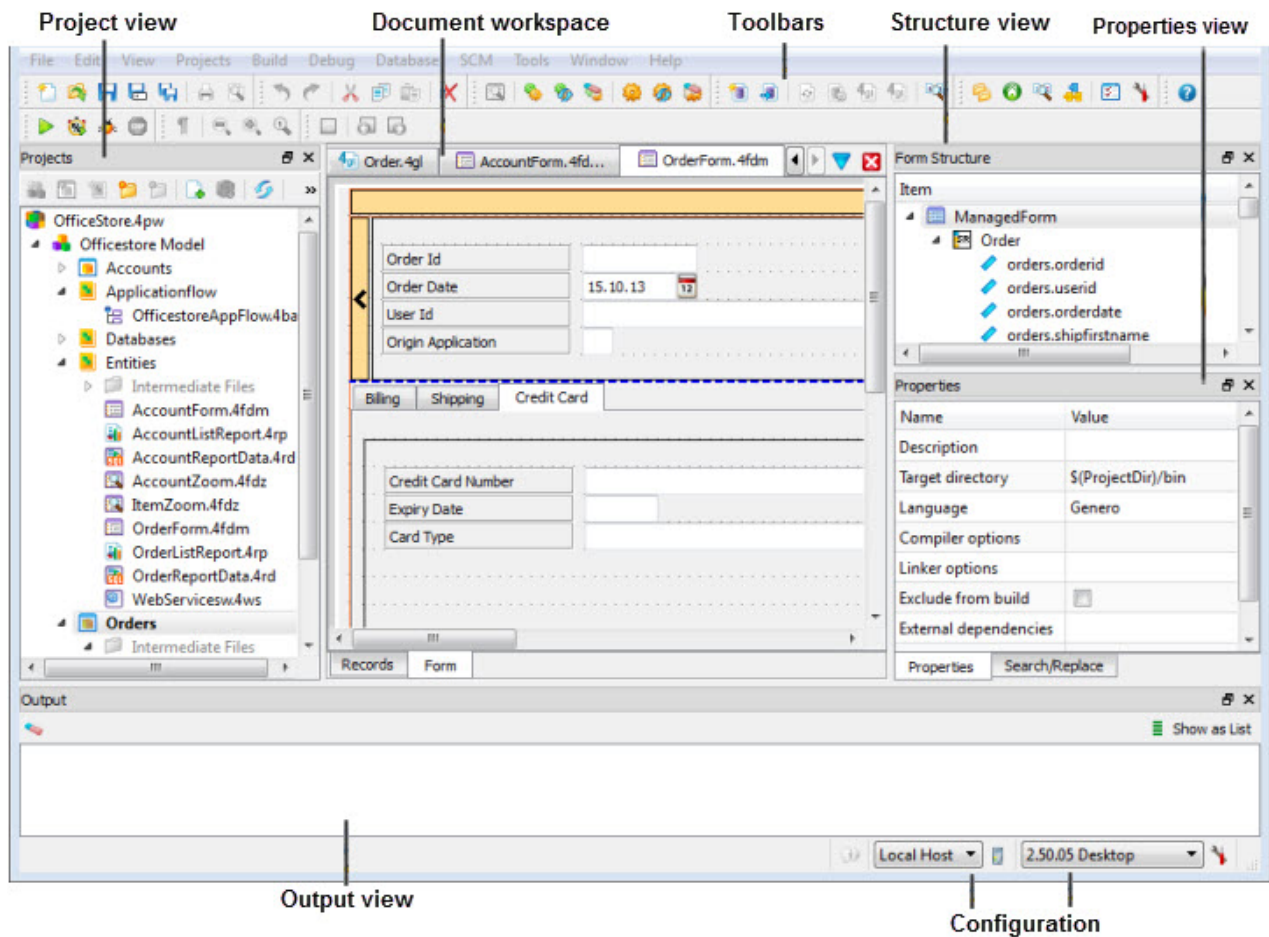


Figure 59: The Genero Studio framework

Tip: In addition to using the Close icon (a red "X"), you can close a tab in the Central Workspace by clicking the mouse wheel on the tab label.

The Welcome Page

The Welcome Page is displayed when Genero Studio is launched. From the Welcome Page, open a recent or sample project or configure the environment or database connection.

Open the Welcome Page by selecting **Tools >> Welcome Page**. The Welcome Page is the first tab in the document workspace.

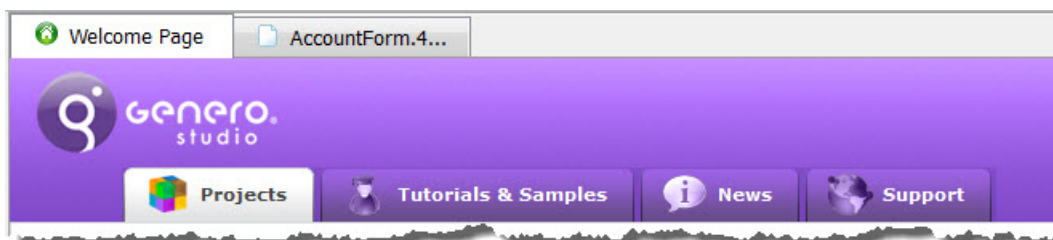


Figure 60: Welcome Page tab

Tabs

Tabs organize the Welcome Page into sections.

Projects	Open a recently opened Project (4pw) or document. See Project Manager on page 340.
Tutorials and Samples	Sample files distributed with Genero Studio; contains links to open the sample projects. See Samples directory on page 103.
News	RSS feed with news about Genero.
Support	Access to support and training videos.

Customizing the Welcome Page

Select **Tools >> Preferences >> Welcome Page** to change what is on the Welcome Page.

Show Welcome Page at start up	Uncheck if you do not wish this page to display.
Template	Select the template for the Welcome Page.
Clear	Clear all stored cookies.
Visible Sections	Check the sections that you wish to be displayed. Enable RSS feeds with the RSS Section checkbox.
Parameters	Use the edit button to edit the feeds used on the page and to add or edit RSS locations.

Modules

Genero Studio is made up of modules managing the design, development, and execution of Genero applications.

A *module* is also commonly referred to as a *plug-in*.

- [BAM - Application Modeling and Code Generation](#) on page 71
- [Code Analyzer](#) on page 72
- [Code Editor](#) on page 75
- [Debugger](#) on page 75
- [Diff](#) on page 76
- [File Browser](#) on page 77
- [Form Designer](#) on page 78
- [Meta-schema Manager](#) on page 79
- [Project Manager](#) on page 80
- [Report Writer](#) on page 81
- [Source Code Management - SVN](#) on page 81

BAM - Application Modeling and Code Generation

Genero Studio Business Application Modeling (BAM) develops business applications from design diagrams rather than from writing code. It automatically generates the logic and source code for a database

application to query, add, update and delete rows in database tables. BAM can generate desktop, web, and mobile applications.

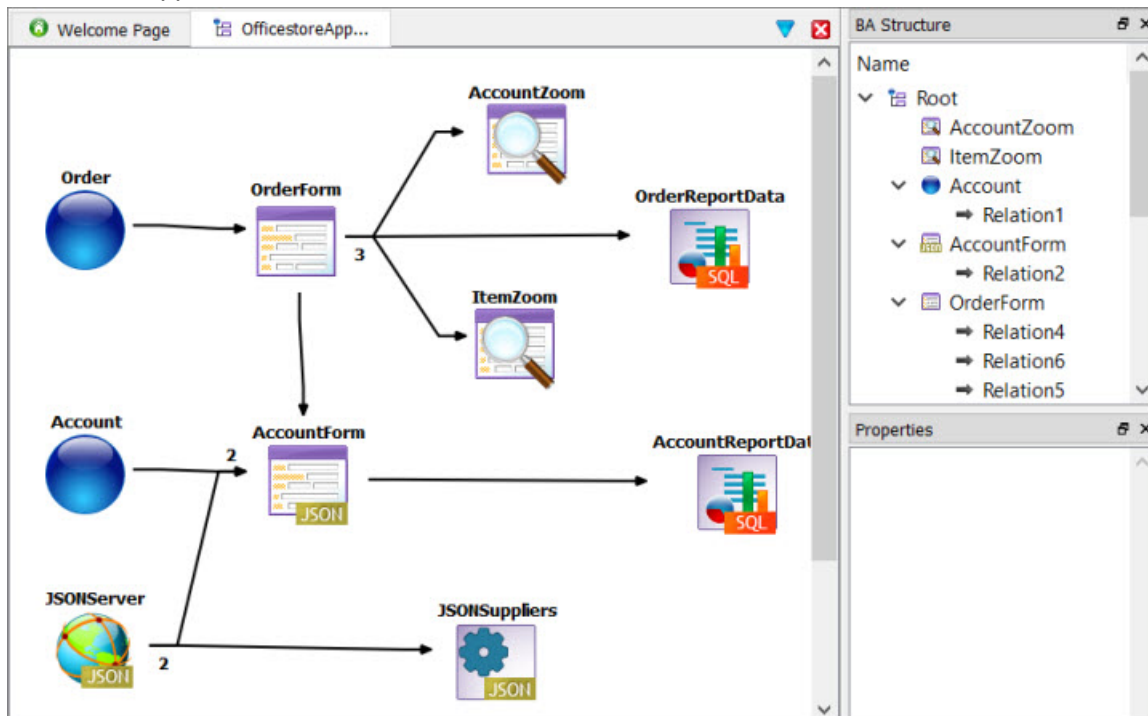


Figure 61: Business Application Modeling (BAM)

Launch by creating or opening a Managed Project from the **File** menu or by opening a new or existing Business Application Model diagram from the **File** menu or Project view.

See [Business Application Modeling \(BAM\)](#) on page 176 for more details.

Code Analyzer

The *Code Analyzer* reverse engineers existing applications and can generate diagrams to provide an overview of the application.

The Sequence Diagram visually displays the flow of your application logic. It shows how the functions of the application call and/or are called by other functions.

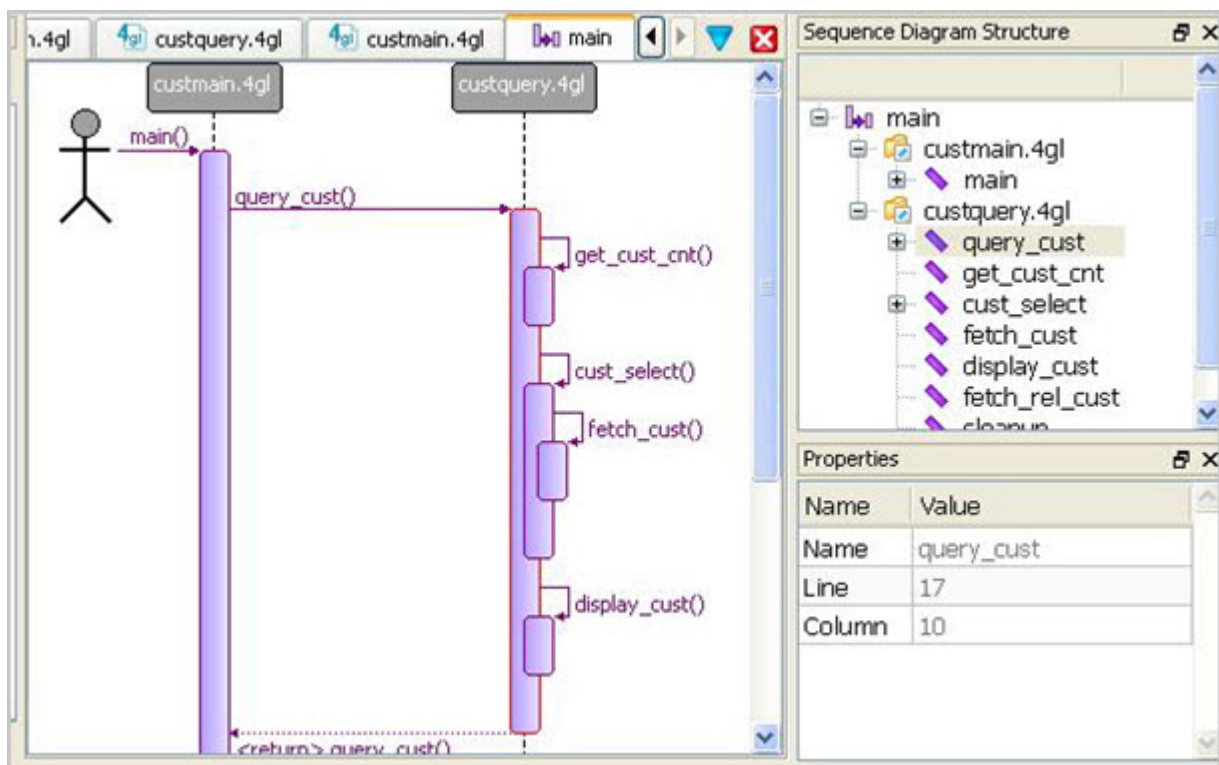


Figure 62: Sequence Diagram

The Dependency Diagram displays a graphical view of the complex relationships between the various pieces of a project. It shows the components that depend on other components, and/or have components that depend on them.

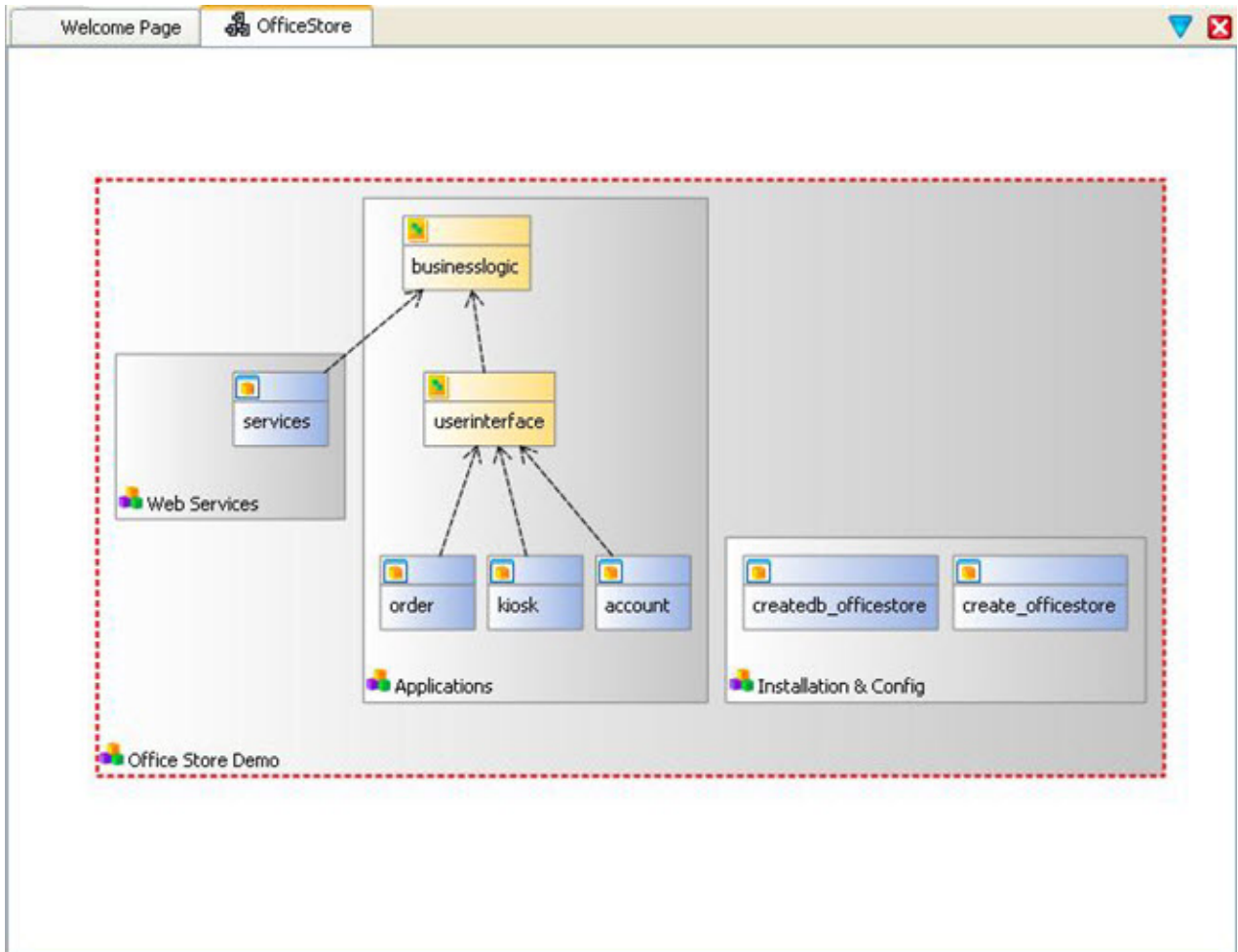


Figure 63: Dependency Diagram

See [Code Analyzer](#) on page 402 for more details.

Code Editor

Code Editor is a programming-oriented editor. In addition to editing source code, it can handle any kind of text or languages such as 4GL and XML. Smart editing features like auto-completion, code templates, text folding, bookmarking, and robust search and replace make coding easier and more efficient.

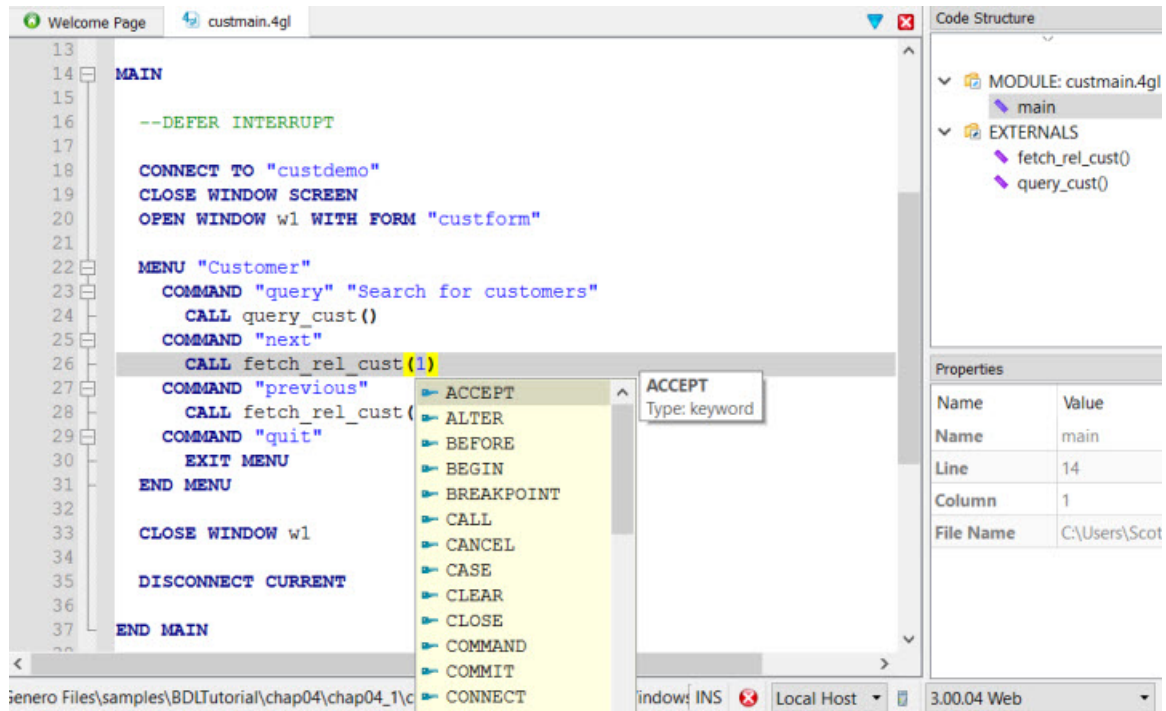


Figure 64: Code Editor

Use **File >> New** or **File >> Open** to open a text file.

See [Code Editor](#) on page 374 for more details.

Debugger

The *Graphical Debugger* provides a graphical interface to test and control the behavior of a Genero application. Navigate through the functions and create and manage breakpoints on functions and code lines. Choose and group together any variables to watch. Follow a number of variables easily, and even alter their values while the application is running, for testing purposes.

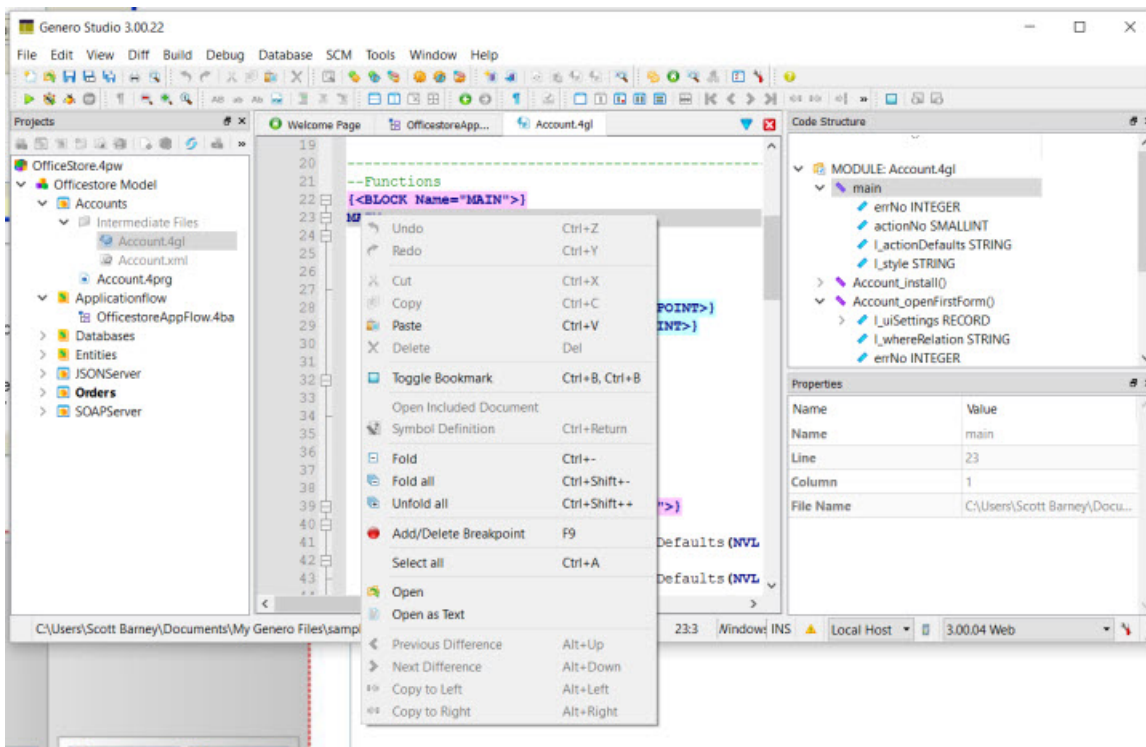


Figure 65: Debugger

Launch with the **Debug** menu options.

See [Graphical Debugger](#) on page 502 for more details.

Diff

The *Diff* tool compares two files: a read-only base copy of the file and a working copy. It is integrated into Code Editor.

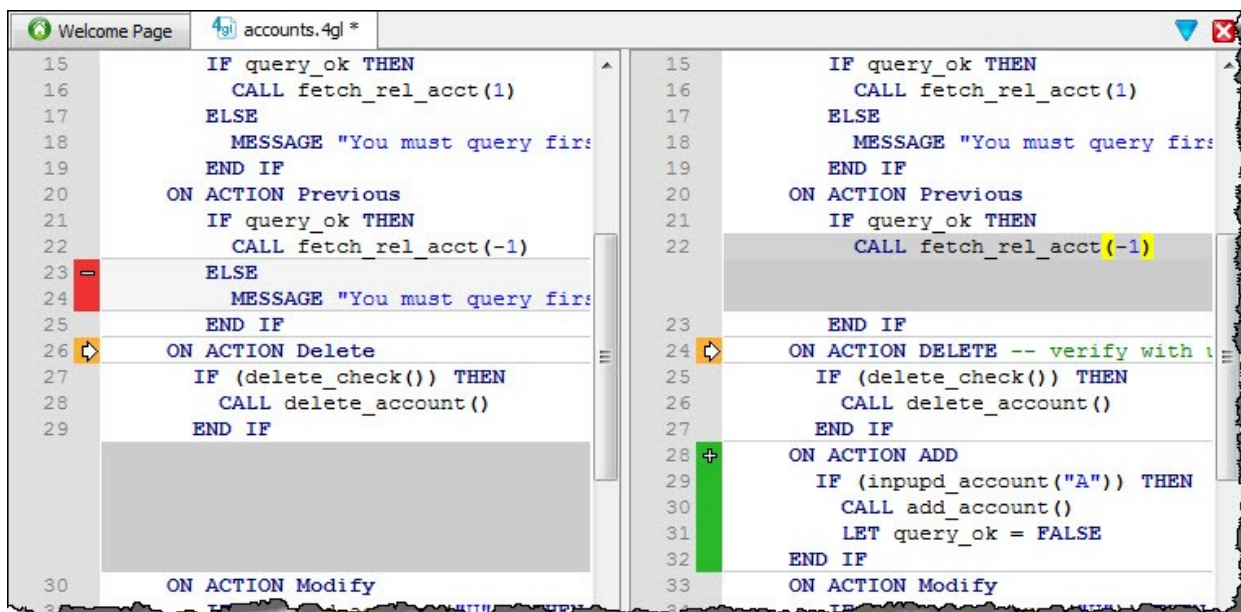


Figure 66: Vertical Dual Diff View display mode

Launch with **Tools >> Diff**.

See [Using the Diff tool](#) on page 380 for more details.

File Browser

File Browser is a tool to navigate, open, delete or rename files, and to create new folders or files on a file system.

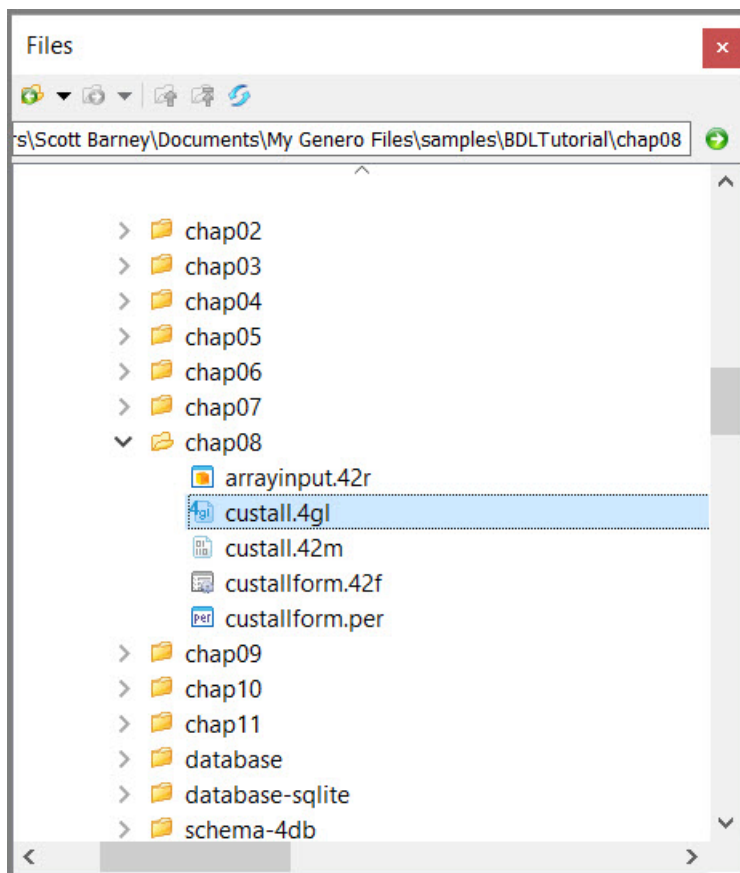


Figure 67: The Files view

Launch with **Tools >> File Browser** or from the Project Manager **Files** tab.

See [File Browser](#) on page 500 for more details.

Form Designer

Form Designer is a visual editor that supports the creation, editing, and layout of Genero forms in Genero Studio.

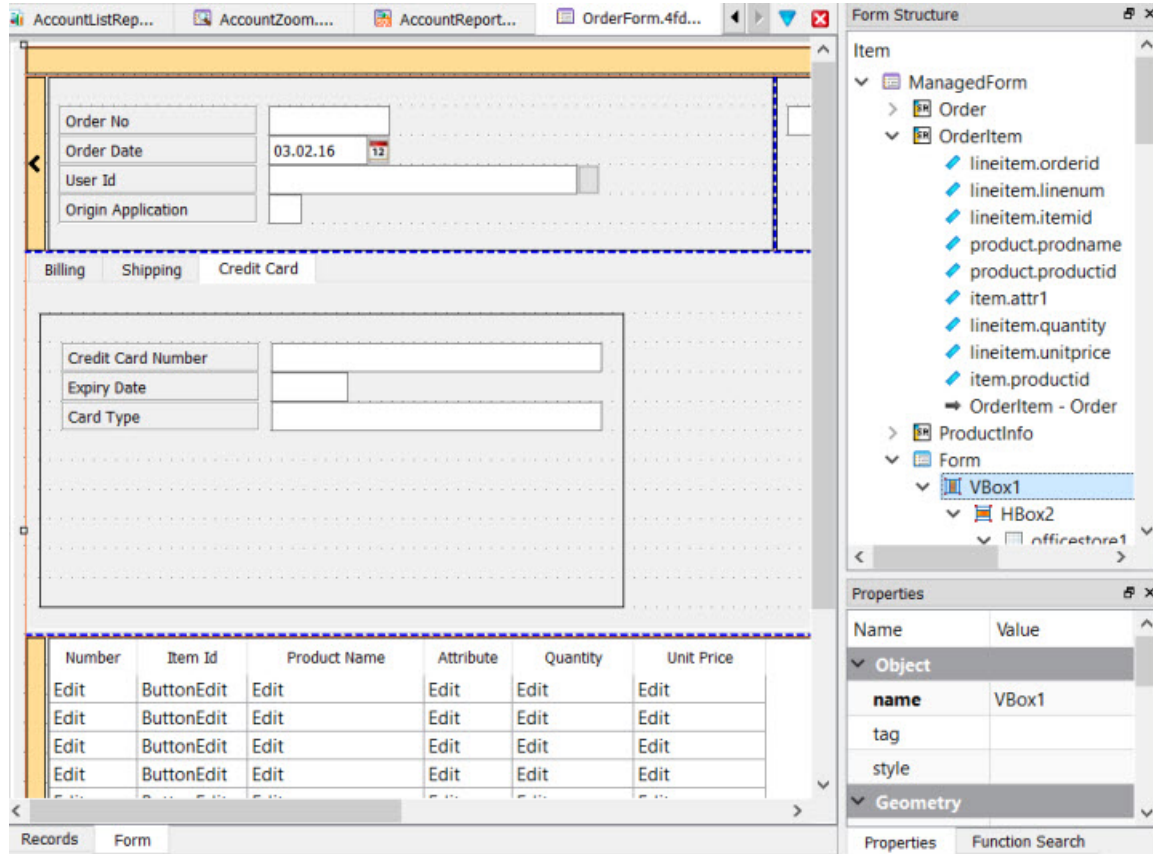


Figure 68: Form Designer

Use **File >> New** or **File >> Open** to create or open a Genero form file.

See [Form Designer](#) on page 406 for more details.

Meta-schema Manager

The *Meta-schema Manager* is a visual tool used to design, create and maintain database meta-schema files.

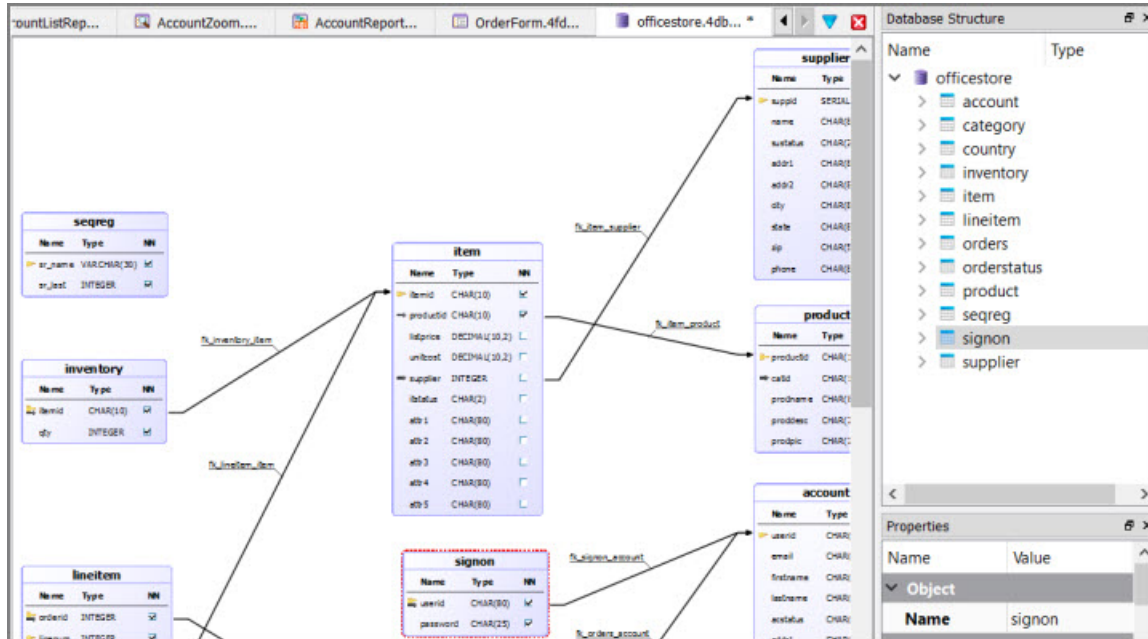


Figure 69: Meta-schema Manager

Use **Database >> New Schema** or **File >> Open** to open an existing schema.

See [Meta-schema Manager](#) on page 288 for more details.

Project Manager

Project Manager is a tool to manage the organization and build of executables from the program's source files.

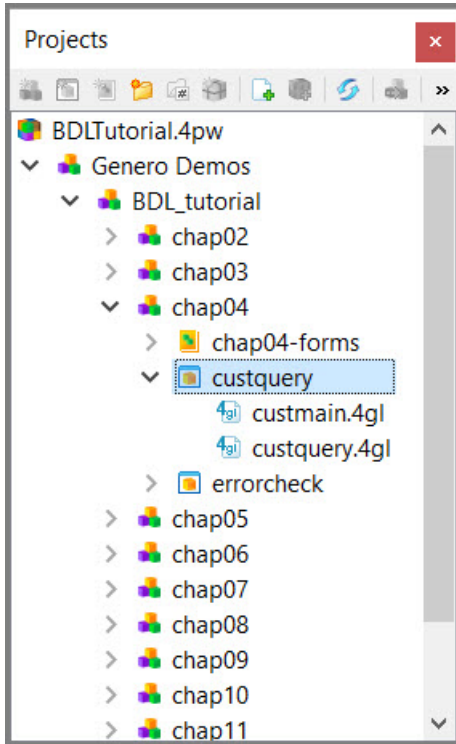


Figure 70: The Projects view

Use **File >> New** or **File >> Open** to open project file.

See [Project Manager](#) on page 340 for more details.

Report Writer

The Genero Report Writer includes a graphical report designer, report engine, and report viewer. Report applications are written using the Genero Business Development Language.

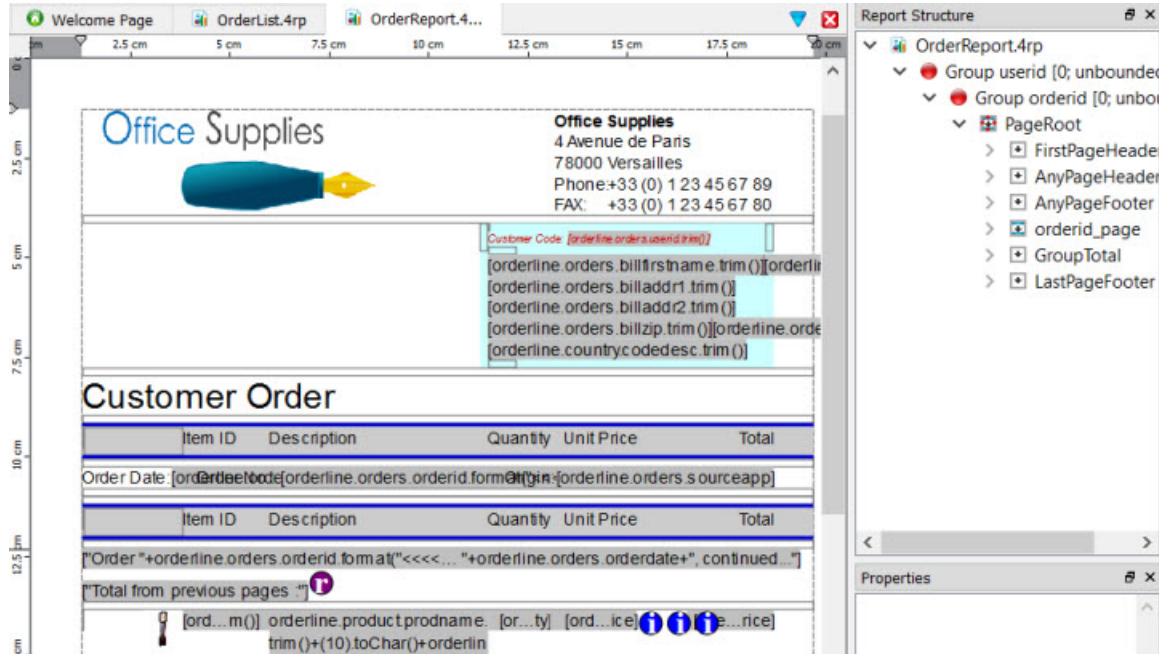


Figure 71: Genero Report Designer

Launch by opening a new or existing Genero Report Design document from the **File** menu or by opening a Genero Report Design document in the Project view.

See [Report Writer](#) on page 551 for more details.

Source Code Management - SVN

Genero Source Code Management (SCM) enables collaborative sharing and maintaining of the files in Genero projects.

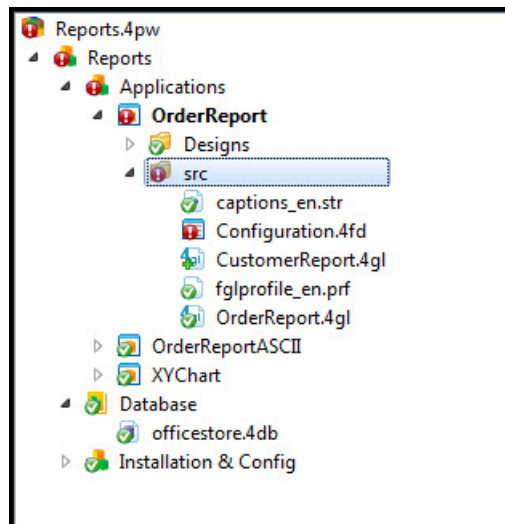


Figure 72: Project view with status icons

See [Source Code Management - SVN](#) on page 529.

Web Services

Web services can be called from a Genero application. The Web Service Wizard guides you through the process of adding a Web service. Web services can also be generated from the Business Application Modeling tools.

See [Web Services](#) on page 910 and [Add Web services \(Server, Services, Forms with services\)](#) on page 215 for more details.

Toolbars and Menus

Menus and Toolbars are constructed dynamically depending on the context and the currently active module.










- [File >> New](#) on page 94
- [Save / Save As / Save All](#) on page 95
- [Opening a file from a prior version](#) on page 96







Customize







Toolbars and menus are fully customizable; you can add, modify, and delete Toolbars, menus and corresponding accelerators using [Tools >> Preferences, User Interface](#). Reorganize the layout of the Toolbar by dragging and dropping a Toolbar to a new location within the Toolbar region or to float on the Genero Studio framework. Right-click to display a context menu for a selected item.


















[Table 14: Toolbars and Menus](#) on page 82 lists the icon, menu option, shortcut key, and description of each option in each Genero Studio menu.



Table 14: Toolbars and Menus







File	File Menu	Accelerator (Shortcut)	Description
	File >> New	Ctrl+N	Create a new file. See File >> New on page 94.
	File >> Open	Ctrl+O	Open a file. See also Opening a file from a prior version on page 96.
	File >> Close file Close Welcome Page	Ctrl+F4	Close open file Welcome Page
	File >> Close project		Close project.
	File >> Checkout...		Checkout from SVN repository. See Checkout files on page 530.
	File >> Save	Ctrl+S	Save current file. See Save / Save As / Save All on page 95.
	File >> Save as...	Ctrl+Alt+S	See Save / Save As / Save All on page 95.
	File >> Save all	Ctrl+Shift+S	Save all unsaved files. See Save / Save As / Save All on page 95.
	File >> Import text form (.per)	Ctrl+I	Import a text form to Form Designer format. See Migrate per file to 4fd on page 411.









File	File Menu	Accelerator (Shortcut)	Description
	File >> Import Project Files		See Import files into the diagram from the project on page 225.
	File >> Export as Image		Exports a diagram to an image file. This option opens a dialog for configuring the image file path, format (such as png and jpg), and size.
	File >> Print ...	Ctrl+P	Print a file. Opens the Print dialog, to print the file using the options available to your operating system printers.
	File >> Print Preview		Preview file. Opens the Print preview dialog. Displays a preview of the printed file in a preview window. For diagrams, scaling and number of pages to use to print can be configured. See Print preview dialog on page 102.
	File >> Poster Printing Setup...		Used to print diagrams, it opens a dialog for configuring the scaling and number of pages to use to print. This action is available from Code Analyzer, Report Writer, Business Application Modeling, and Meta-schema Manager.
	File >> Recent documents		List of recent opened documents.
	File >> Recent projects		List of recently opened projects
	File >> Quit		Quit Genero Studio












Edit	Edit Menu	Accelerator (Shortcut)	Description
	Edit >> Undo	Ctrl+Z	Undo the last action(s).
	Edit >> Redo	Ctrl+Y	Redo the last action(s).
	Edit >> Cut	Ctrl+X	Cut to clipboard.
	Edit >> Copy	Ctrl+C	Copy to clipboard.
	Edit >> Paste	Ctrl+V	Paste from clipboard.
	Edit >> Delete	Del	Delete item.
	Edit >> Select All	Ctrl+A	Select all on area with focus.





Edit	Edit Menu	Accelerator (Shortcut)	Description
	Edit >> Search/Replace		See Search and replace on page 377.
	Edit >> Search/Replace >> Find	CTRL+F	Find in current file.
	Edit >> Search/Replace >> Find in Files	CTRL+SHIFT+F	Find in selected files.
	Edit >> Search/Replace >> Find Previous	SHIFT+F3	Find previous occurrence.
	Edit >> Search/Replace >> Find Next	F3	Find next occurrence.
	Edit >> Search/Replace >> Replace previous		Replace previous occurrence.
	Edit >> Search/Replace >> Replace next		Replace next occurrence.
	Edit >> Search/Replace >> Replace all		Replace all occurrences.
	Edit >> Search/Replace >> Stop search		Stop current search
	Edit >> Search/Replace >> Clear all Search Results		Clear all search results.
	Edit >> Bookmarks >> Toggle Bookmark	CTRL+B, CTRL+B	Toggle bookmark at location. See Bookmarks on page 376.
	Edit >> Bookmarks >> Previous Bookmark	CTRL+B, CTRL+P	Go to previous bookmark.
	Edit >> Bookmarks >> Next Bookmark	CTRL+B, CTRL+N	Go to next bookmark.
	Edit >> Bookmarks >> Delete all Bookmarks	CTRL+B, CTRL+L	Go to next bookmark.
	Edit >> Open Line	Ctrl+Shift+L	Open a new line.
	Edit >> Delete Line/Word		Delete current word or line.
	Edit >> Selection		Select current word or line. See Selection keymap on page 395.
	Edit >> Go To	Ctrl+G	Go to location in file.
	Edit >> Comment >> Toggle Line Comment	Ctrl+K, Ctrl+K	Toggle line comment.
	Edit >> Comment >> Block Comment	Ctrl+K, Ctrl+C	Comment section of code.
	Edit >> Comment >> Block Uncomment	Ctrl+K, Ctrl+U	Uncomment section of code.
	Edit >> Completion >> Auto Complete	Ctrl+Space	Completes a line of code or prompts for a valid keyword in the syntax. See Auto










Edit	Edit Menu	Accelerator (Shortcut)	Description
	Edit >> Completion >> Use Template	Ctrl+T	completion (Ctrl+Space) on page 376. Select template from list. Code templates (Ctrl+T) on page 376.
AB	Edit >> Case >> Upper Case	Alt+Shift+U	Change selection to upper case.
ab	Edit >> Case >> Lower Case	Alt+Shift+L	Change selection to lower case.
	Edit >> Case >> Uppercase Word	Ctrl+Alt+Shift+W	Change word to upper case.
	Edit >> Case >> Lowercase Word	Ctrl+Alt+W	Change word to lower case.
Ab	Edit >> Case >> Toggle Case	Ctrl+U	Toggle case.
	Edit >> Case >> Convert Keywords to Uppercase		Convert keywords to upper case.
	Edit >> Strip Trailing Spaces	Ctrl+Shift+T	Delete all spaces at the end of lines.
	Edit >> Format and Indent	Ctrl+Shift+I	Format the document by indenting XML elements and removing extra spaces. This option is enabled only for XML documents.
	Edit >> Convert to		Convert text to Windows, UNIX, or Mac format.
	Edit >>VI Editing Mode	Ctrl+Shift+E	Use VI commands to edit the file. See vi emulator and diff tools in Code Editor on page 51 .




View	View Menu	Accelerator (Shortcut)	Description
	View >> Show characters >> Show all Characters		Show all characters in document.
	View >> Show characters >> Show Line Ending Characters		Show line ending characters in document.
	View >> Show characters >> Show White Spaces		Show white spaces in document.
	View >> Zoom out		Zoom out.
	View >> Zoom in		Zoom in.
	View >> Actual size		Restore default zoom.




View	View Menu	Accelerator (Shortcut)	Description
	View >> Split Horizontal	Ctrl+Alt+H	Split document horizontally. See Split a document on page 377
	View >> Split Vertical	Ctrl+Alt+V	Split document vertically. See Split a document on page 377
	View >> Unsplit	Ctrl+Alt+Shift+N	Unsplit.
	View >> Unsplit all	Ctrl+Alt+Shift+A	Unsplit all.
	View >> Fold	Ctrl+-	Fold section of text. See Fold text on page 375.
	View >> Unfold	Ctrl++	Unfold section of text.
	View >> Fold all	Ctrl+Shift+-	Fold all text. See Fold text on page 375.
	View >> Unfold all	Ctrl+Shift++	Unfold all text.
	View >> Languages		Specify the language that Code Editor uses for formatting and error marking.









Diff	Diff Menu	Accelerator (Shortcut)	Description
	Diff >> Normal View		Normal view editing mode.
	Diff >> Diff View		View diff in single pane. See Using the Diff tool on page 380.
	Diff >> Diff View with Deleted Blocks		Diff view with deleted blocks.
	Diff >> Vertical Dual Diff View		View diff in dual vertical panes.
	Diff >> Horizontal Dual Diff View		View diff in dual horizontal panes.
	Diff >> Base File		Select base file to use for diff.
	Diff >> Flip Sides		Flip sides.
	Diff >> First Difference	Alt+Home	Go to first difference.
	Diff >> Previous Difference	Alt+Up	Go to previous difference.
	Diff >> Next Difference	Alt+Down	Go to next difference.
	Diff >> Last Difference	Alt+End	Go to last difference.
	Diff >> Copy to Right	Alt+Right	Copy change to right.












Diff	Diff Menu	Accelerator (Shortcut)	Description
	Diff >> Copy to Left	Alt+Left	Copy change to left.
	Diff >> Copy All to Right		Copy all changes to right.
	Diff >> Copy All to Left		Copy all changes to left.
	Diff >> Create Patch		Create patch.















Build	Build Menu	Accelerator (Shortcut)	Description
	Build >> Compile File	F6	Compile selected file.
	Build >> Preview	Ctrl+Shift+P	Preview the selected form.
	Build >> Build all	F8	Build all of project.
	Build >> Rebuild all	Shift+F8	Rebuild all of project. Clean and then build files that are not up-to-date.
	Build >> Clean all		Clean all of selected. Erase all output files defined in the Build and Link rules.
	Build >> Build	F7	Build default application. Compile and link files in the default application.
	Build >> Rebuild	Shift+F7	Rebuild selected or default application.
	Build >> Clean		Clean selected or default application.
	Build >> Abort Last Task	Ctrl+Shift+F5	Abort last started task.





Debug	Debug Menu	Accelerator (Shortcut)	Description
	Debug >> Execute	Ctrl+F5	Execute selected or default application.
	Debug >> Execute with Profiler		Execute selected or default application with Profiler. See Profiler on page 510.
	Debug >> Debug	F5	Debug selected or default application. See Graphical Debugger on page 502.
	Debug >> Attach to Process		Debug a running application. See Start the Debugger on a running program on page 502.
	Debug >> Attach to Mobile Process		Debug a mobile application. See Debug a mobile application on page 504 .



Debug	Debug Menu	Accelerator (Shortcut)	Description
	Debug >> Add/Delete Breakpoint	F9	Add or delete a breakpoint at current location.
	Debug >> Enable/Disable Breakpoint	Ctrl+F9	Enable or disable breakpoint at current location.
	Debug >> Abort Last Task		Abort last started task.



Database	Database Menu	Accelerator (Shortcut)	Description
	Database >> Extract Schema...	Ctrl+M	Extract schema from database.
	Database >> Import SCH File...		Import schema from sch file.
	Database >> Update Schema		Update database meta-schema file from database. See Update a meta-schema from database on page 302.
	Database >> Generate SCH File		Generate a BDL schema file (<code>sch</code>) from the meta-schema file. This file is automatically created by Genero Studio when a meta-schema is compiled. See BDL schema file (sch) on page 293.
	Database >> Generate Database Creation Script		Used to generate a source 4gl file that can be used to create a new database and tables according to the meta-schema file. See Generate a database script from meta-schema on page 303.
	Database >> Generate Database Update Script		Used to generate a source 4gl file that can be used to update an existing database based on the meta-schema file. See Generate a database script from meta-schema on page 303.
	Database >> Generate Schema Documentation		Generates an HTML page with details on the meta-schema. See Generate meta-schema documentation on page 305.
	Database >> Diff Schema...		See Comparing two meta-schemas on page 302.

Database	Database Menu	Accelerator (Shortcut)	Description
	Database >> Add Table		See Add new tables and columns on page 294.
	Database >> Add Column		See Add new tables and columns on page 294.
	Database >> Add Constraint or Index		See Add constraints or indexes on page 294.
	Database >> Add Foreign Key		See Add foreign keys on page 295.
	Database >> Select	Esc	Selection tool.
	Database >> Edit	F2	Edit properties of the item selected.
	Database >> Insert Column Before	Ctrl+Shift+Ins	See Add new tables and columns on page 294.
	Database >> Insert Column After	Ins	See Add new tables and columns on page 294.
	Database >> Revert		See Revert schema changes dialog on page 315.
	Database >> Layout		Rearrange the items in the diagram.
	Database >> Filter Items		The Filter View dialog allows you to hide and show items on a diagram.
	Database >> Locate in Diagram		This action brings focus in the diagram to the selected item. If the selected object is not visible in the current view, the Meta-schema Manager will try to find another view where the object is visible. If no view is found, you are prompted to make the object visible in the current view or to create a new view.
	Database >> Advanced Properties		Specify the extraction and/or generation options. See Extract meta-schema information from database on page 228 and Generate a database script from meta-schema on page 303.
SCM	SCM Menu	Accelerator (Shortcut)	Description
	SCM >> Checkout...		Checkout files from a repository. See Checkout files on page 530.

SCM	SCM Menu	Accelerator (Shortcut)	Description
	SCM >> Show Log		Display the SVN Log view. See SVN Log view on page 541.
	SCM >> Review Changes		Display the SVN Status view. See Commit / Review changes on page 531.
	SCM >> Show locks		Display SVN Locks view. See SVN Locks view on page 541.
	SCM >> Merge/Revert...		Display the SVN Merge/Revert dialog. See Merge and revert on page 535.
	SCM >> Copy...		Display the SVN Copy dialog. See Copy working files and directories on page 535.
	SCM >> Switch...		Display the SVN Switch dialog. See Move a working copy (Switch) on page 536.
	SCM >> Apply patch...		The Apply patch command applies a patch file to files in the repository. See Apply patch on page 536.
	SCM >> Browse Repository		Display the SVN Repository Browser . See Browse repository on page 537.
	SCM >> Update		Display the SVN Update dialog. See Update / Update All on page 534.
	SCM >> Lock		Locking a file provides exclusive rights to a user for changing that file in the repository. See Locking on page 532.
	SCM >> Cleanup		The Cleanup command cleans up the working copy, removing stale locks.
	SCM >> Blame		The Blame command shows author and revision information inline for specified files or URLs.
	SCM >> Blame...		Display the SVN Blame dialog.
	SCM >> Properties		Display the SVN Properties dialog. See SVN Properties dialog on page 548.


Tools	Tools Menu	Accelerator (Shortcut)	Description
	Tools >> Welcome Page		Open Welcome Page. See The Welcome Page on page 70.
	Tools >> File Browser		Open File Browser. See File Browser on page 500.
	Tools >> Diff		Open Diff tool. See Using the Diff tool on page 380.
	Tools >> Dependency Diagram		Open Dependency Diagram. See Dependency Diagrams on page 403.
	Tools >> Genero Tools		Access to Genero tools such as the BDL Licenser and the Genero Workplace Window.
	Tools >> Android Tools >> Auto-configure Android SDK		Run the scripts to auto-configure the Android SDK for use in Genero mobile development. See <i>Genero Mobile Developer Guide</i> .
	Tools >> Android Tools >> Create Android Virtual Device (x86)		Create the default Android emulator. See <i>Genero Mobile Developer Guide</i> .
	Tools >> Android Tools >> Launch Android Emulator (x86)		Launch the default Android emulator. See <i>Genero Mobile Developer Guide</i> .
	Tools >> Android Tools >> Deploy Genero Mobile for Android		Deploy the GMA client to the connected device or emulator. See <i>Genero Mobile Developer Guide</i> .
	Tools >> Android Tools >> Open Android SDK Manager		Launch the installed Android SDK Manager.
	Tools >> Android Tools >> Open Android Debug Monitor		Launch Android Debug Monitor tool which is part of the Android SDK.
	Tools >> Android Tools >> Open Android Virtual Devices Manager		Launch the Android Virtual Devices Manager which is a part of the Android SDK.
	Tools >> Android Tools >> List Devices		Lists each attached device and its unique id.
	Tools >> Android Tools >> Show Android Device Logs		Displays the Android Device Logs to the Output view. See <i>Viewing the device logs</i> in the <i>Genero Mobile Developer Guide</i> .
	Tools >> Android Tools >> Display Standard output and error		Display or stop display of the Standard output and errors to the Output view. See <i>Viewing the program</i>

Tools	Tools Menu	Accelerator (Shortcut)	Description
	Tools >> Android Tools >> Stop display Standard output and error		<i>logs</i> in the <i>Genero Mobile Developer Guide</i> .
	Tools >> Android Tools >> Show AUI Tree		Display AUI tree in browser. See <i>Viewing the AUI Tree</i> in the <i>Genero Mobile Developer Guide</i> .
	Tools >> iOS Tools >> Launch iOS Simulator		Launch the default iOS Simulator. See <i>Genero Mobile Developer Guide</i> .)
	Tools >> iOS Tools >> Deploy Genero Mobile for iOS		Deploy the GMI client to the connected device or simulator. See <i>Genero Mobile Developer Guide</i> .
	Tools >> iOS Tools >> iOS App Store deployment >> Open iTunesConnect		Open the https://itunesconnect.apple.com webpage.
	Tools >> iOS Tools >> iOS App Store deployment >> Launch Application Loader		Opens Xcode's "Application Loader" application.
	Tools >> iOS Tools >> List Devices		Lists each attached device and its unique id.
	Tools >> iOS Tools >> Show AUI Tree		Display AUI tree in browser. See <i>Viewing the AUI Tree</i> in the <i>Genero Mobile Developer Guide</i> .
	Tools >> Current Config >> Launch GDC	Ctrl+Shift+G	Launch the Genero Desktop Client (GDC).
	Tools >> Current Config >> Launch GDC...		Prompt for GDC options before launching GDC.
	Tools >> Current Config >> Open Application Server Monitor		Open Application Server Monitor.
	Tools >> Current Config >> Genero BDL Licenser		Open Genero BDL Licenser program.
	Tools >> Current Config >> Genero Report Engine Licenser		Open Genero Report Engine Licenser program.
	Tools >> Current Config >> Genero Workplace Window		Open Genero Workplace Window.
	Tools >> Global Setup >> Edit File Associations		See File associations configuration on page 106.
	Tools >> Global Setup >> Edit Build Rules		Edit global build rules. See What are build rules on page 344.

Tools	Tools Menu	Accelerator (Shortcut)	Description
	Tools >> Global Setup >> Edit Package Rules		Edit global package rules. See Platform: Package and deploy rules on page 1003.
	Tools >> Specific Setup >> Edit File Associations		See File associations configuration on page 106.
	Tools >> Specific Setup >> Edit Build Rules		See What are build rules on page 344.
	Tools >> Specific Setup >> Reload	Ctrl+Shift+R	Reload Application Generator settings.
	Tools >> Specific Setup >> Clean orphan properties		Opening a file generated with a template version different than the one set in Application Generator preferences may produce a warning indicating that some properties are not found in the current template definition.
	Tools >> Server Connections		Connect to a remote server. See Setting up a remote environment on page 154
	Tools >> Genero Configurations	Ctrl+Shift+C	Open Configuration Management to modify configurations. See Genero Configuration Management dialog on page 171.
	Tools >> Preferences	Alt+Shift+F12	Open Preferences. See Setting Preferences on page 106.
	Tools >> Preferences, Code Editor, XML Schema/DTD		Assign external schema file.
	Tools>>Translation		Change the language of the labels in Genero Studio.

Window	Window Menu	Accelerator (Shortcut)	Description
	Window >> Close all other Documents		Close all documents except the current document.
	Window >> Close all Documents	Ctrl+Shift+F4	Close all open documents.
	Window >> Views		Select a view to show in the framework.
	Window >> Workspaces		Select a workspace type. See Workspaces configuration on page 108
	Window >> Toggle Document Editing	Alt+F11	Toggle to/from Document Editing workspace.

Window	Window Menu	Accelerator (Shortcut)	Description
	Window >> Toggle Full Screen	Alt+Shift+F11	Toggle to/from full screen display.

Help	Help Menu	Accelerator (Shortcut)	Description
	Help >> Help	F1	Open Help.

File >> New

`File >> New` creates a new file. When created, the corresponding Genero Studio component is opened. For example, Code Editor is launched for new `4g1` files.

Table 15: File >> New options

Category	Section	Item Type
Design (Model Driven Architecture)	Project	<ul style="list-style-type: none"> Managed Project (<code>4pw</code>) - project for a generated application
	Application Modeling	<ul style="list-style-type: none"> Business Application diagram (<code>4ba</code>) - application design diagram, used in application generation
	Database	<ul style="list-style-type: none"> Schema (<code>4dbx</code>) - database schema Schema from Database (<code>4dbx</code>) - extracts schema from database
	Managed Code	<ul style="list-style-type: none"> CRUD Form (<code>4fdm</code>) - create an empty CRUD (Create, Read, Update, Delete) form CRUD Form from Database (<code>4fdm</code>) - create a CRUD form generated from database tables Report data (<code>4rd</code>) - create empty business record for generated report Report data from database (<code>4rd</code>) - create business record for generated report from database tables Zoom Form (<code>4fdz</code>) - create an empty zoom form Zoom Form from Database (<code>4fdz</code>) - zoom form generated from a database tables
	Resources	<ul style="list-style-type: none"> Action defaults (<code>4fd</code>) - a Genero action defaults file Toolbar (<code>4tb</code>) - a Genero Toolbar definition file Topmenu (<code>4tm</code>) - a Genero Topmenu definition file Style (<code>4st</code>) - a Genero Style definition file
Genero Files	Project	<ul style="list-style-type: none"> Simple project (<code>4pw</code>) Deploy project

Category	Section	Item Type
	Database	<ul style="list-style-type: none"> • Schema (4db) - database schema • Schema from Database (4db) - extracts schema from database
	Sources	<ul style="list-style-type: none"> • Source (4gl) - a Genero BDL source code module • Form (4fd) - a Genero Studio form definition file • Form as text (per) - a Genero BDL form definition file in text format • Form from database (4fd) - calls the Form from Table wizard
	Resources	<ul style="list-style-type: none"> • Action defaults (4fd) - a Genero action defaults file • Toolbar (4tb) - a Genero Toolbar definition file • Topmenu (4tm) - a Genero Topmenu definition file • Style (4st) - a Genero Style definition file • Startmenu (4sm) - A Genero Start Menu definition file • Localized String (str) - Genero Localized String file
Other files	New Files	<ul style="list-style-type: none"> • With no extension • Text (txt) • XML (xml)
Reports	Report Data	<ul style="list-style-type: none"> • Report from database (4gl) - to generate BDL code to extract data from database
	Report Designs	<ul style="list-style-type: none"> • Empty report (4rp) - a blank report design document • List report (4rp) - template of a report design document formatted for a list
Web / AS	Sources	<ul style="list-style-type: none"> • HTML (html) • CSS (css) • XHTML (xhtml) • JavaScript™ (js)
	Configuration	<ul style="list-style-type: none"> • Application Configuration (xcf)
SOA	Services	<ul style="list-style-type: none"> • Soap Server (4gl) - create a 4gl application and server stub from wsdl file

Save / Save As / Save All

When you save a file for the first time, or select **File >> Save As**, the **Save as** dialog opens.

If a project is open, you can also choose to create a link to the file in a group node or virtual folder.

Path Specify where the file should be saved in the file system. Click the magnifying glass icon to browse for the path and enter the file name.

Insert the file in the project If checked, you can choose the application or library node, or a virtual folder, from the list that is displayed in the dialog. Use the plus icons to expand the list.

If a project is not open, the file can be saved in the file system only. No link can be created in a group node.

Save in Select the path where the file should be saved in the file system.

Filename Enter the name of the file.

Save as type Change the type of the file, if desired.

If you have previously saved the file, choosing **File >> Save** saves it again with the same parameters. No dialog opens.

The **Save All** option saves the project and its contents.

Opening a file from a prior version

If you open a file that had been saved with a prior version, you have the option to convert and open the file in the current version or to open in a different version of Genero Studio.

Open in new instance Displays a selector to find the version of Genero Studio associated with the file version.

Open Converts the file in memory and opens it. If saved, it will be saved to the latest file format.

Cancel Cancels opening the file.

Views

Views are the panels in the Genero Studio framework that display information about the current document or project.

To view a list of views, select **Window >> Views**.

- [Show, dock, or move a view](#) on page 96
- [Views Listing](#) on page 97

Show, dock, or move a view

Views (panels) can be hidden and shown, docked or undocked, and moved within the framework.

Show or hide a view

Show or hide views by right-clicking in the Genero Studio window title, or use **Window >> Views**.

Note: The Data View and Tool Box views are only listed when you have a report design document open in the Document view.

For a list of hot keys to show and hide the views, look at **Window >> Views**. Where available, the hot key is listed next to the view name.

Select visible views

To open the **Manage Views** dialog, select **Window >> Views >> Manage Views ...**. A dialog opens listing all views. Use the checkbox to set each view as visible or hidden.

- If the view has a check in its checkbox, then the view is shown in Genero Studio.
- If the view does not have a check in its checkbox, then the view is hidden.

To make all views visible, select the **Select all** checkbox located at the bottom of the dialog.

Dock or undock a view

Undock a view by double-clicking on its title bar. Re-dock a view to its last position by double-clicking on its title bar.

Move views

Move a view to by selecting its title bar and dragging it to float or to a new position in the framework. As you move the view, shaded areas appear showing you valid locations to place the view.

Views Listing

This is a list of views available in Genero Studio.

- Bookmarks - See [Bookmarks view](#) on page 97.
- Data View - See [Adding report data \(Data view\)](#) on page 666. This view is only available when using Genero Report Designer.
- DB Explorer - See [DB Explorer](#) on page 332.
- DB Schemas - See [DB Schemas tab](#) on page 316.
- Document - The [Document view](#) is the default view, also known as the central work area. You cannot hide the Document view.
- Document Errors - See [Document Errors view](#) on page 98.
- Files - See [File Browser](#) on page 500
- Function Search - see [Function search](#) on page 379.
- Output - See [Output view](#) on page 98.
- Projects - See [Projects view](#) on page 99.
- Properties - See [Properties view](#) on page 99
- SVN Locks - See [SVN Locks view](#) on page 541.
- SVN Log - See [SVN Log view](#) on page 541.
- SVN Repository - See [Browse repository](#) on page 537.
- SVN Status - See [SVN Status view](#) on page 543.
- Search Results - See [The Search Results view](#) on page 393.
- Search/Replace - See [The Search/Replace view](#) on page 391.
- Structure - See [Structure view](#) on page 100.
- Tasks - See [Tasks view](#) on page 100.
- Tool Box - See [The Tool Box view](#) on page 661. This view is only available when using Genero Report Designer.

Bookmarks view

The Bookmarks view lists all the bookmarks that have been added to documents in Genero Studio.

You can navigate, add, or remove bookmarks from the Bookmarks view.

The integrated Toolbar and right-click context menu include these options:

Toggle bookmark	Adds or removes a bookmark in the current document, provided the current module supports bookmarks.
Previous / Next bookmark	Activates the previous / next bookmark in the view.
Remove bookmark / Remove all bookmarks	Removes current or all defined bookmarks.

Document view

The Document view is the area designated for working on the document of a Genero Studio module, a form in Form Designer, for example. It is commonly called the *central work area*.

Each document has its own tabbed page. The Document view may contain as many tabbed documents as needed.

- Click a tab to **bring a document to the front**.
- Click the Select a desired document icon



at the top right corner to **select a desired document** from the drop-down list.

- Click the close icon



to **close** the current document.

- Right-click a tab to display this menu:
 - Close the **current document**
 - Close **all other documents**
 - Close **all documents**

The Window menu allows you to display various views, providing additional information.

Document Errors view

The Document Errors view displays errors related to a document.

- Select the error number and press the F1 key to display additional information about the error.
- Use [Tools>>Preferences](#), [Genero Studio Preferences](#), [Messages](#) to hide a specific information or warning message.
- View `BUG` or `TODO` notations found in your code. Enter the notation into your code files with `--KEYWORD <message>` where `KEYWORD` is `BUG` or `TODO`. After compiling, put focus in the Project Manager view to see these messages in the Document Errors tab.

Output view

The Output view displays messages related to the output and errors specific to the process being performed.

The **Filter Messages** checkbox shows error messages, warning messages, and/or information messages.

Properties view

Select an item in the structure view or central work area to display its properties in the Properties view; the property values can be viewed and/or changed.

Projects view

The Projects view displays a project and its components.

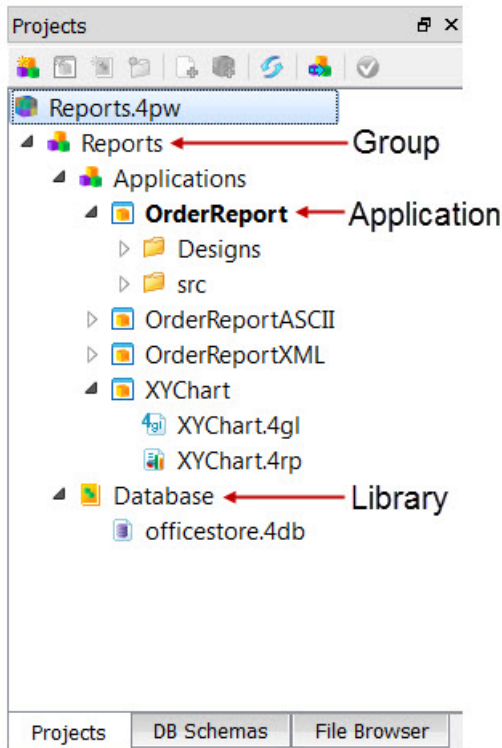


Figure 73: Projects View

Right-click on a node to display a context menu of available actions.

Select a node to set its properties in the [Project Manager node properties](#) on page 364.

Table 16: Projects View Nodes

Node	Description
<i>projectName.4pw</i>	The active project (4pw) is the root node.
Group	Contains the Application and Library nodes that make up the project. Properties defined for the Group node can be inherited by the Application and Library nodes.
Application	Generates an executable program (42x). It can contain both files and virtual folders. Only one of the files may have a MAIN statement: One application node = one executable. The name of the application node is used as the name of the 42x file, so it must be unique and can only contain characters allowed by the file system. The default application is shown in boldface. Use the Toolbar icon to set a different application as the default. The options on the Build menu execute for the default application.
Library	Used to group binary files into a single library and generate a library file (42x). It can contain both files and virtual folders. The name of the library node is used as the name of the 42x file, so it must be unique and can only contain characters allowed by the file system. Libraries should be used when creating

Node	Description
	<p>a set of features having a common goal, like the logic of an application, a library of mathematical functions, etc. A library can also be used to group other project files together (images, styles or other resources). If a library node contains no 4g1 file, no 42x is built.</p> <p>A library from a different project can be added to a project using the right-click menu option Add external dependencies.</p> <p>Important: A library must be linked to any application in which it will be used by right-clicking the application node and selecting Advanced Properties, dependencies. The checkbox for any required library must be checked.</p>
Folder	<p>Folders are virtual folders only, providing a way to group the source files within an application or library node. Folders can contain both files and other folders.</p>
File	<p>A link to the file in the file system that has the same name as the node. Renaming a File node also renames the file that is stored on the disk. Project Manager will accept any type of file. Opening a file opens the corresponding Genero Studio module, such as Code Editor. If there is no corresponding Genero Studio module, it asks the operating system to open it.</p>

Structure view

The Structure View displays a tree showing the structure of the current document, a form or source code file, for example.

Tasks view

The Tasks view displays the completion status of current tasks and includes action to abort tasks if needed.

Dialogs

Dialogs display as the result of a selected action. In most cases, you must complete the dialog to continue.

- [Filter View dialog](#) on page 100
- [Print preview dialog](#) on page 102

Filter View dialog

The Filter View dialog allows you to hide and show items on a diagram. Right-click the diagram or an item on the diagram and select the **Filter Items** option. Check boxes on the Filter View dialog allow you to specify which items to display.

To access the Filter view, right-click on a diagram in:

- Meta-schema Manager
- Dependency Diagram
- Business Application Modeling

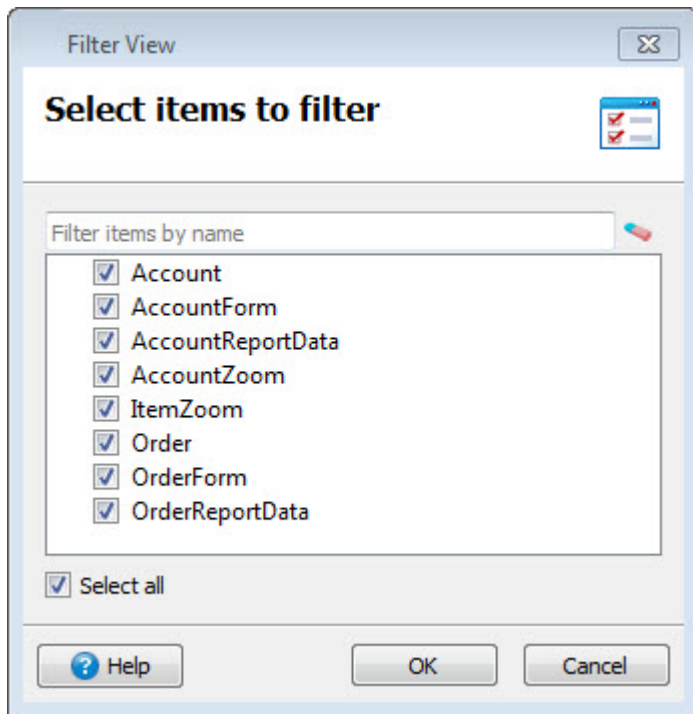


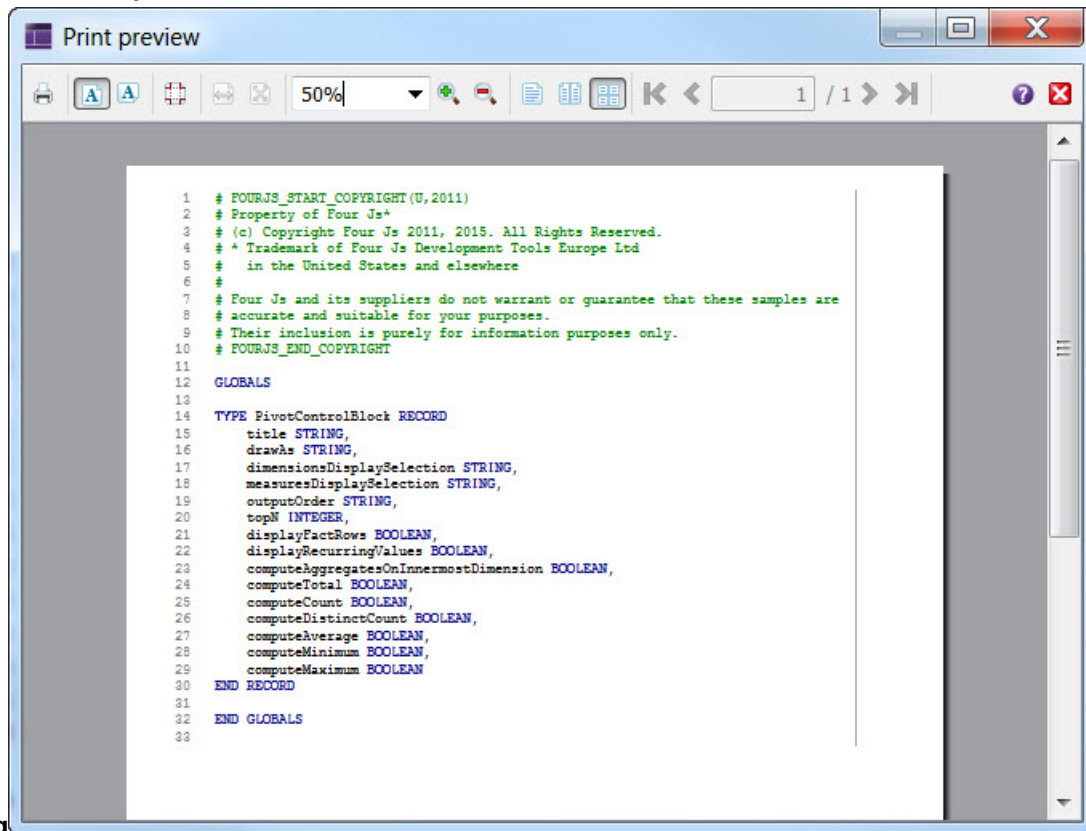
Figure 74: Filter View

Print preview dialog

The **Print preview** dialog allows you to preview a document or diagram, before sending it to the printer. Various toolbar icons allow you to change how the document or diagram prints.

The **Print preview** dialog appears when you select **File >> Print Preview...**

Figure 75: Print preview



dialog

Actions available from the Print preview dialog are presented as icons in the toolbar.

Table 17: Print preview toolbar icons

Icon Name	Description
Print	Opens the print dialog.
Portrait printing	Formats layout to be a portrait page.
Landscape printing	Formats layout to be a landscape page.
Page setup	Opens the page setup dialog.
Fit to width	Fits the content to the width of the page.
Fit to page	Fits the content to the size of the page.
Percentage combobox	Sets the display size in proportion of the final print size.
Zoom in	Enlarges the display percentage.
Zoom out	Reduces the display percentage.
Single page	View a single page at a time.
Facing pages	View pages in book style, with even and odd pages facing each other.

Icon Name	Description
Overview of pages	View all pages in two columns.
Page navigator	Navigate to the front of the print document, to the previous page, to a specific page, to the next page, or to the last page of the print document.
Help	Access the help documentation.
Page navigator	Close the dialog.

Learning to use Genero Studio

Reference topics for the samples directory, setting preferences, accessing help and more.

- [Samples directory](#) on page 103
- [Integrating existing applications](#) on page 105
- [Setting Preferences](#) on page 106
- [Access Help](#) on page 113

Command line options

Genero Studio can be launched at the command line with the command `generostudio` at a console, terminal, or using the Genero Workplace Window.

Syntax

```
generostudio [options]
```

Table 18: Genero Studio command line options

Option	Description
<code>-h</code>	Display help instead of the standard behavior.
<code>-v</code>	Display program name and version information.
<code>-height HEIGHT</code>	Set window height in pixels.
<code>-width WIDTH</code>	Set window width in pixels
<code>-translate LOCALE</code>	Set translation file where <i>LOCALE</i> is <i>zh_TW</i> , the locale for Taiwanese (using traditional Chinese characters) or <i>zh_CN</i> , the locale for Chinese (using simplified Chinese characters).
<code>-log [all, req, <id>]</code>	Set the log level. Use <i>all</i> to turn on all logs, <i>req</i> to turn on server request logs, and <i><id></i> to turn on logs for a given module.
<code>-diff FILE1 FILE2</code>	Open the given files in diff mode. See Using the Diff tool on page 380.

Samples directory

The samples directory contains demo files, programs, and databases.

The default location for the sample files is `My Documents/My Genero Files/samples`.

Genero Mobile samples are located in `My Documents/My Genero Mobile Files/samples`. See *Genero Mobile demos* in the *Genero Mobile User Guide* for more information.

HelloWorld	Contains the <code>HelloWorld.4pw</code> project, which has the source code for the very simple Hello World application.
OfficeStore	Contains the <code>OfficeStore.4pw</code> project which is an application generated using Business Application Modeling (BAM). It uses the officestore sample database.
WebComponentChart	Contains the <code>WebComponentChart.4pw</code> project, which has the source code for including a charting Web Component within a Genero application.
HTML5ClientTutorial	Contains the <code>HTML5ClientTutorial</code> project files, which demonstrate customization of a web application.
Reports	Contains the <code>Reports.4pw</code> project, which has a set of example programs that create reports using Genero Report Writer.
DSConfig	Contains the <code>DSConfig.4pw</code> project, which has the program source code to complete an <code>FGLPROFILE</code> configuration file with the necessary information for a database selected by the user. An <code>FGLPROFILE</code> file already exists for the sample databases, but you must create one if you choose to use your own database.
databases	Contains directories for the sample SQLite databases custdemo and officestore . The directories also contain schema files and a program file with the necessary SQL commands to recreate the databases for your own database software, if desired. These files are grouped in project files, <code>OfficeStoreSetup.4pw</code> and <code>CustDemoSetup.4pw</code> .
BDL tutorial	Contains the <code>BDLTutorial.4pw</code> project, which has a set of tutorial programs that illustrate the use of the Genero Business Development Language (BDL), to be used in conjunction with the BDL Tutorial.

The necessary data for the sample programs is stored in the provided SQLite databases.

Open the project for one of the samples to examine and execute the sample programs. From the [Genero Studio Welcome Page](#), select the **Tutorials and Samples** tab. Links to open the sample projects are displayed under **Samples and Demos**.

Integrating existing applications

Considerations for integrating existing applications into Genero Studio.

Table 19: Considerations for integrating existing applications

Consideration	Information
Determine whether you will use a local or remote configuration of Genero Studio.	See Software configuration scenarios on page 115.
Incorporate a schema.	See Create a meta-schema on page 227.
Import your files.	See Import existing files as a new project on page 341.
Reorganize your project nodes and check dependencies	See Default organization of imported files .
Start correcting errors - syntax, link errors, language errors.	See Code Editor basics on page 374.
Recompile and link. Adapt makefiles.	Adapt makefiles or create build rule that calls a makefile. See Building and linking programs on page 343.
Run your application in traditional mode.	<p>You can use the <i>traditional GUI mode</i> to ease migration from TUI based applications to GUI mode.</p> <p>With the traditional mode, application windows bound to forms using a <code>SCREEN</code> section will be displayed as simple boxes in a main front end window. Other windows bound to forms defined with the <code>LAYOUT</code> section will be displayed a new GUI windows.</p> <p>The traditional GUI mode can be enabled with an <code>FGLPROFILE</code> entry:</p> <pre>gui.uiMode = "traditional"</pre> <p>By default, the traditional GUI mode is off.</p>
Consider form migration from <code>per</code> to <code>4fd</code> and begin using Form Designer.	See Migrate per file to 4fd on page 411, Creating the user interface on page 407, Form Designer usage on page 440.
Modernize forms.	See Creating the user interface on page 407.
Analyze code.	See Code Analyzer on page 402.
Debug.	See Graphical Debugger on page 502.
Switch client configurations from desktop to web.	See Change the active configuration on page 117.
Improve the meta-schema.	See Adding more information to a meta-schema on page 293
Consider modernizing code.	See www.4js.com
Consider report strategy.	See Report Writer on page 551.

Consideration	Information
Customize studio environment.	See Integrate your tools into Genero Studio on page 113, General Preferences on page 106.

Setting Preferences

Customize Genero Studio to meet your needs.

- [General Preferences](#) on page 106
- [User interface preferences](#) on page 110
- [Compiler and Runtime preferences](#) on page 112
- [Integrate your tools into Genero Studio](#) on page 113

Select **Tools >> Preferences** to modify the behavior of Genero Studio modules.

Save, test, or cancel changes you have made.

Load from default	Reloads the initial default configuration.
OK	Save and apply all modifications, then exit Preferences window.
Cancel	Undo all modifications and exit the Preferences window. The last saved values are restored.
Apply	Confirms your updates, allowing you to test the new configuration. (No save is performed.) If you want this configuration to become permanent, save it by pressing the OK button.

General Preferences

Select **Tools >> Preferences, General** to access these preferences.

Table 20: General preferences

Preference	Description
Text File Encoding	See Language support (text encoding) on page 163.
Proxy Setting	If you need to access a web service using a proxy, enter: <ul style="list-style-type: none"> • Host - the hostname or host IP address • Port - The port number on which the service is listening
Browser Setting	You can specify the browser to be used with the Genero Web Client: <ul style="list-style-type: none"> • Use default web browser • Use specific web browser - enter the filename of the executable, including path, of the browser that you wish to use, or select the file in your file system using the Browse icon.
Slow Networks	Select to optimize for slow networks.
Users and Passwords	Press the Clear button to clear the passwords stored in Genero Studio.

File associations configuration

Select **Tools >> Global setup >> Edit File Associations** or **Tools >> Specific setup >> Edit File Associations** to edit the standard or specific file associations, such as for a template.

The template file associations are available if the *GSTSETUPDIR* environment variable is set and refers to an BAM template directory. The dialog lists the file associations set in the file-types.xml file found in the template directory.

You can associate file types handled by Genero Studio modules with Genero Studio predefined actions, or with [User Actions](#) that you have defined.

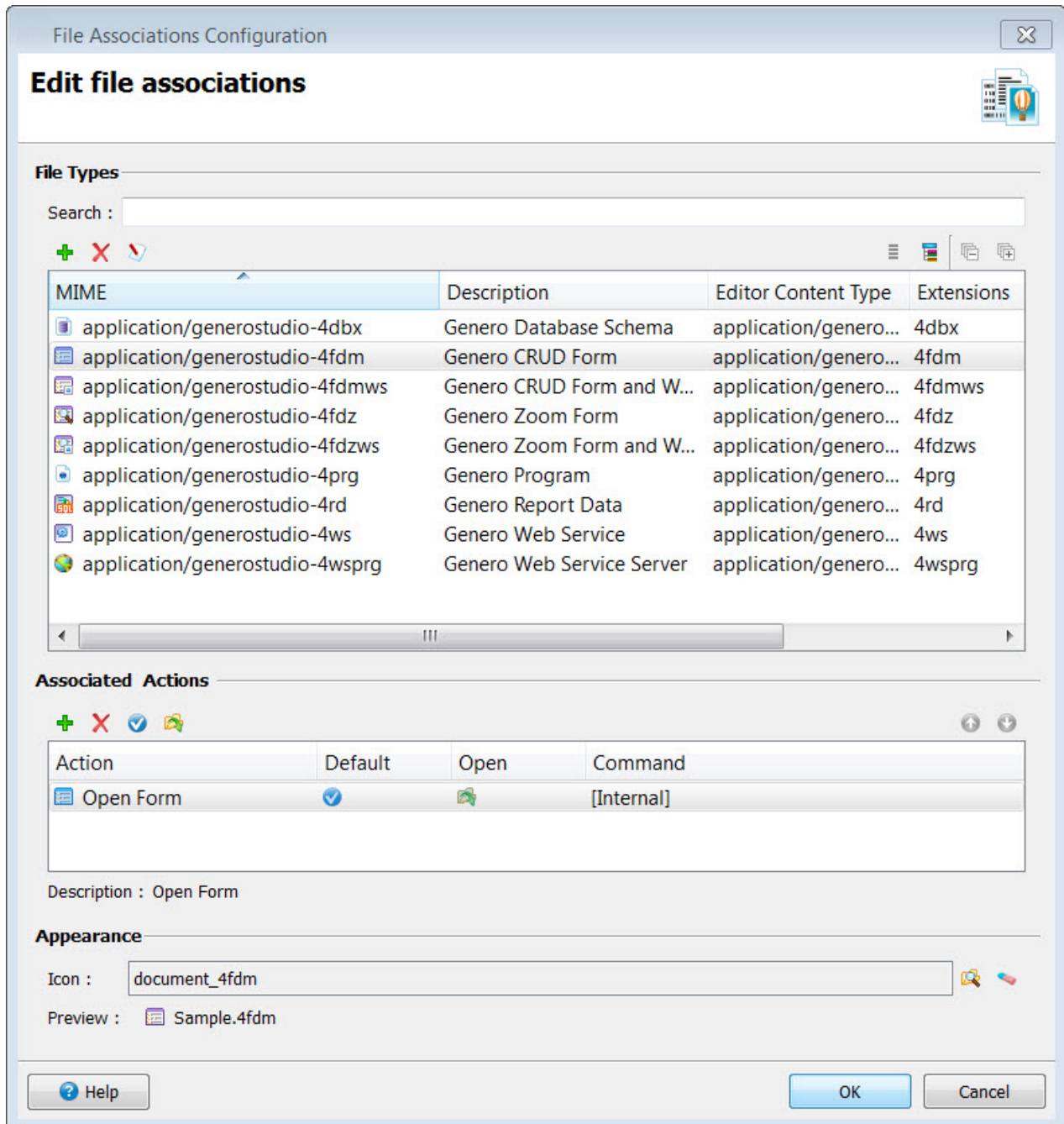


Figure 76: File Associations

Use the integrated Toolbar to add, modify or delete a new file type.

File Types

Search

Search for the file type you wish to view or edit.

Mime	Predefined file type/identifier
Description	Optional description.
Editor Content Type	Type of editor used with this file type.
Extensions	File extensions matching the file type.
Filenames	Optional file names matching the file type.

Associated Actions

When you select **Add** from the icons at the top of the Actions list, the available actions are displayed in a list for your selection. [User Actions](#) must be defined using **Preferences, User Actions** before they will appear in the list.

Action	Predefined action names.
Default	Indicates the action is the default; use the icon to change the default.
Open	Indicates the action is used to open the file; use the icon to change.
Command	Predefined command to be executed; may be [Internal].

Appearance

Icon	Search for the icon on your file system. The path is specified for icons that are not in the default Genero Studio icons directory. See Image directory structure on page 952.
Preview	Sample is displayed.

Workspaces configuration

Open last session documents on startup	If checked, the last documents that you had open will be restored when you start the next session.
---	--

Workspaces

Normal	Displays the usual views - Project, Output, Structure, Properties.
Debug	Removes some of the normal views to focus on the Debugger; automatically switches the workspace when starting and stopping a debug session. Keeps track of the current workspace and the one used while debugging, and switches between them when a debug session is started or stopped.
Document	Displays the current document and the Project Manager to focus on document editing.
User	Contains the user workspaces; those that have been duplicated. Only User workspaces may be renamed or deleted.

User actions configuration

User actions allow you to define commands that can be invoked within Genero Studio.

Once a user action is defined, you can add it to the [Toolbar configuration](#) on page 111 and [Menus configuration](#) on page 111

Existing User Actions, if any, are displayed in a list:

Action	Names of existing user actions
Command Line	Command line for the corresponding action

Use the Search box to find a user action in the list.

Use the integrated Toolbar to add, edit, or delete a user action.

The screenshot shows a dialog box for configuring a user action. It contains the following fields and values:

- Name :** UserAction1
- Label :** SQLite Studio
- Description :** Launch SQLite Studio
- Icon :** question
- Command line :** "C:\Program Files\SQLiteStudio\sqlitestudio-2.0.21.exe"

At the bottom, there is a Verbose checkbox and an **Insert Variable** button.

Figure 77: User Action dialog

Name	Name of user action.
Label	Label for user action as it will be displayed in Genero Studio action listing, menus, or Toolbar.
Description	Optional description.
Icon	Optional icon.
Command line	Command to be executed, in quotes if command includes spaces. Arguments can be included on the command line, as needed. Use the Insert Variable button to select from a list of predefined variables that can be used in the command, or you can create a variable to prompt the user to input an argument:

- This example would execute the program **stest** using the selected file.

```
C:\myprograms\stest $(FileName)
```

- This example would execute the program **stest** prompting the user to input the file name as value for the user defined variable (*\$p1*)

```
C:\myprograms\stest $(p1)
```

Once you have created a new user action, you can:

- [Associate it with files](#) of a specific mime type.

- [Add it to a Toolbar.](#)
- [Add it to a menu.](#)

Messages configuration

Message dialogs

A list of messages is displayed as a tree, grouped by Genero Studio module. Expand a module listing to display its messages.

You can prevent a message from displaying while you are using Genero Studio by unchecking the **Show** checkbox for the message.

Use the **Search** box to locate a specific message.

Document Error Filters

This list specifies the message id of information or warning messages that you wish to be hidden in the Document Errors tab of the Output View. Use the icons to add and remove message ids from the list.

History configuration

In the work area of many of the Genero Studio modules, the names of previously used items are stored and displayed as history. Genero Studio allows you to manage these History lists.

History list

The name of each list that you can manage is displayed. These are the only lists that you can alter.

History Details

Table 21: History Details

Property	Description
History Size	Specify the maximum number of items displayed in a list.
Allow Duplicates	If this box is checked, duplicate names can be added to the list of items. If the list contains duplicate items, unchecking the box will cause a dialog box to be displayed, asking you to confirm the removal of the duplicate items from the list.

History items

Icons allow you to:

- **Add** - add a new item
- **Delete** - remove the selected item
- **Edit** - change the name of the selected item
- **Up/Down arrow** - Rearranges the items in the list by moving a selected item up and down.
- **Default** - specifies the default item for the selected list.

User interface preferences

Set preferences for toolbars, menu bars, and accelerators.

- [Toolbars configuration](#)
- [Menubars configuration](#)
- [Accelerators configuration](#)

Toolbar configuration

Set preferences for toolbar configuration.

Expand a Toolbar listed in the tree to display its actions.

Select a Toolbar or action, and use the icons or right-click to display the menu options:

- Add a new Toolbar
- Add an action to a Toolbar
- Remove or rename a Toolbar
- Remove or rename an action
- Add or remove a separator

Item order within a Toolbar may be changed using "drag and drop".

Menus configuration

Set preferences for menu configuration.

Expand a menu listed in the tree to display its actions.

Select a menu or action, and use the icons or right-click to display the menu options to:

- Add a new menu at the same level, or a submenu
- Add an action to a menu
- Remove or rename a menu
- Remove or rename an action
- Add or remove a separator

Item order may be changed using "drag and drop". This applies to menu order within the tool bar tree, and to icon order within one menu.

Accelerators (Shortcuts) configuration

Accelerators and their associated Genero Studio actions are defined in a default set called a profile.

Profile

Profiles correspond to a set of accelerators for each action. They are saved in Accelerator Profile files having an extension of `.apr`. A **default** profile file is preinstalled in the Genero Studio installation directory. Initially, this is the only profile available, and it cannot be modified. You can create your own profile by duplicating the **default** profile and modifying the accelerators associated with an action. All the accelerators (except menu accelerators) are disabled during editing, and are re-enabled when editing is completed or you leave the Accelerators configuration window.

- **current** - This combobox displays the currently active profile, with the names of other available profiles displayed in a dropdown list. The **default** profile, which cannot be modified, has a lock icon. To change profiles, select a profile from the dropdown list, and click **OK**.
- **Duplicate** - Creates a new profile. The **Duplicate Profile dialog** allows you to enter the name of a new profile, which will be a copy of the currently active profile. The file is saved in this user directory:

Documents and Settings\<username>\Application Data\<companyname>\Genero Studio<version>.

Modifications in the accelerators can be made in your new user profile. All the user profiles that you create will have a user icon associated with their name in the list in the **current** combobox.

- **Remove** - Removes the currently active profile. This removes the associated `.apr` file. The **default** profile provided by Genero Studio cannot be removed.

Accelerator profiles can be shared with other users:

- **Import** - Imports an `.apr` file. An Open dialog allows you to browse for the profile file. When you click **OK**, the selected file is copied into the user directory and appears in the current combobox.
- **Export** - Exports an `.apr` file. A Save As dialog allows you to save your profile file.

Actions

The actions and the associated accelerators for a particular profile are displayed in a table, grouped by application by default. Expand the application group to display all its actions. The **Search** box allows you to locate a specific action in the tree.

You can sort the table to make the resolution of conflicts easier. Use the icons to switch between viewing the tree grouped by application, or in alphabetical order by action. To sort the tree by the accelerator column, click the column title.

Accelerators

The accelerators that are defined for an action display in a list. Use the icons or right-click for a menu of options to:

- **Add** a new, or additional, accelerator to an action using the Add New Accelerator dialog
- **Edit** the accelerator for an action using the Edit Accelerator dialog
- **Remove** the selected accelerator from an action
- **Set the selected accelerator as the default** for an action

Add New Accelerator/Edit Accelerator dialog

Enter your own key combination for the accelerator by hitting the keys in order; use the **Tab** button to enter a Tab key.

Use the **Backspace** key to erase the latest keypress, or the **Clear** button to erase the entire key combination.

Compiler and Runtime preferences

Set preferences for the compiler and runtime.

- [Run/debug configuration](#)
- [Compilation configuration](#)

Run/debug configuration

Set preferences for runtime and debug configuration.

Console

Clear output before each launch

Specify whether to clear the output after each launch.

Logging

Enable proxy logging

Enable proxy logging.

Compilation configuration

Set preferences for compilation configuration

General

Maximum number of parallel compilation tasks

Specify the maximum number of parallel compilations that you want to allow. although only one link at a time can be done, multiple file nodes can be compiled simultaneously. This speeds up the build process if the machine has multiple processors.

If you specify 0 maximum, Project Manager uses the number of processors available on the local machine (i.e. 2 for a dual core machine).

Maximum number of erroneous files before stop compiling

Specify the maximum number of files containing errors to be permitted before compilation is automatically stopped.

Compute dependencies between files (recommended)

This enables the computation of dependencies. They will be computed when the project is loaded, and is enabled by default. It can be useful to disable it for very large projects or projects on network drives with performance problems. However, it should ordinarily be enabled, in order for the build system to take the #include and globals into account.

Compute additional information about functions

Recommended.

Save modified files before compile and build

Specify whether to automatically save files before compile and build process.

Verbose mode for build/link/execution rules

Specify to display build, link, and execution rule commands in output.

Output

Clear output before each compilation

When selected, clears the Output view before each new compilation.

Integrate your tools into Genero Studio

To call your tools and scripts directly from Genero Studio, you can customize global menus and Toolbars and also the context menus that are used in the Project Manager and File Browser.

You can set up your commands to ask for parameters (using input variables) or use the preset parameters proposed by the Genero Studio environment (current file, current directory, and so on).

All commands run in the current environment set by the Genero Configuration or the Project. The working directory is set to the location of the file to which the command applies.

Access Help

Help is available within Genero Studio by selecting the **Help >> Help** menu option, the



icon, or the **F1** key.

Creating with Quick Starts

A quick start provides you with simple step-by-step instructions for completing specific tasks.

Quick starts are located in sections where they have supporting topics. Here is a list of all the quick starts that can be found in the *Genero Studio User Guide*.

Getting Started with Genero Studio

[Quick Start: Tour of Genero Studio](#) on page 41

Business Application Modeling (BAM)

[Quick Start: Generate an application](#) on page 176

[Quick Start: Generating a mobile app](#) on page 182

Project Manager

[Quick Start: Create a project](#) on page 340

Form Designer

[Quick Start: Creating a first form](#) on page 407

BAM Template Developer Guide

[Quick Start: Customizing templates](#) on page 930

Configuring Genero Studio

Configure Genero Studio to best meet your needs.

- [Software configuration scenarios](#) on page 115
- [Default configuration](#) on page 117
- [Setting up a local environment](#) on page 139
- [Setting up a remote environment](#) on page 154
- [Share projects / source code management](#) on page 162
- [Access a database](#) on page 163
- [Language support \(text encoding\)](#) on page 163
- [Compiler / Runtime configuration \(Genero Installations\)](#) on page 172
- [Environment sets](#) on page 140
- [Desktop: GDC configurations](#) on page 146
- [Web: GAS/GWC configurations](#) on page 147
- [Command line options](#) on page 103
- [Configuration reference](#) on page 170

Software configuration scenarios

Genero Studio can be installed with all or some Genero components and for a local or remote environment.

Installations

When you install the Genero Studio software package, you can automatically install these Genero components:

- Genero Business Development Language (BDL)
- Genero Desktop Client (GDC) - to be used for display
- Genero Web Client (GWC) - to be used for display
- Genero Web Services (GWS)
- Genero Application Server (GAS)

When you install the Genero Mobile software package, you install:

- Genero Studio
- Genero Business Development Language (BDL)
- Genero Mobile for Android (GMA), and/or Genero Mobile for iOS (GMI)

See the Genero Mobile User Guide for setup and configuration of Genero Mobile.

Local Environments

In a local Genero Studio environment, developers work individually, with a complete Genero suite stack installed for each developer working on his local machine. The database can be on developers local machines, or accessed from remote servers. Some group development can be enabled through the use of an external Version Control System.

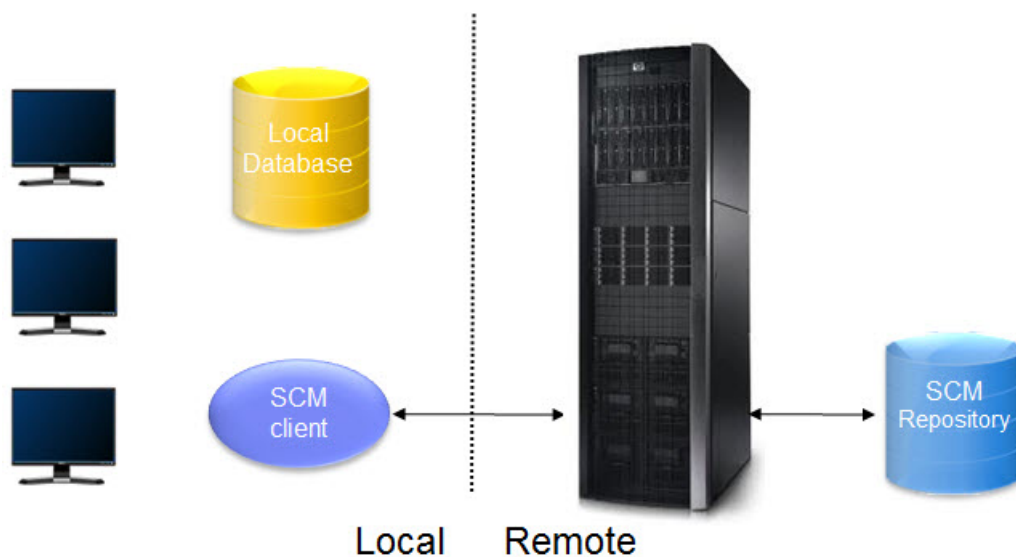


Figure 78: One or Multiple Users local database

In a local environment with multiple developers, the Genero Studio, Genero DVM, and front-end client software (GDC/GWC) is on the local machines. These machines connect to database servers and source code on remote machines.

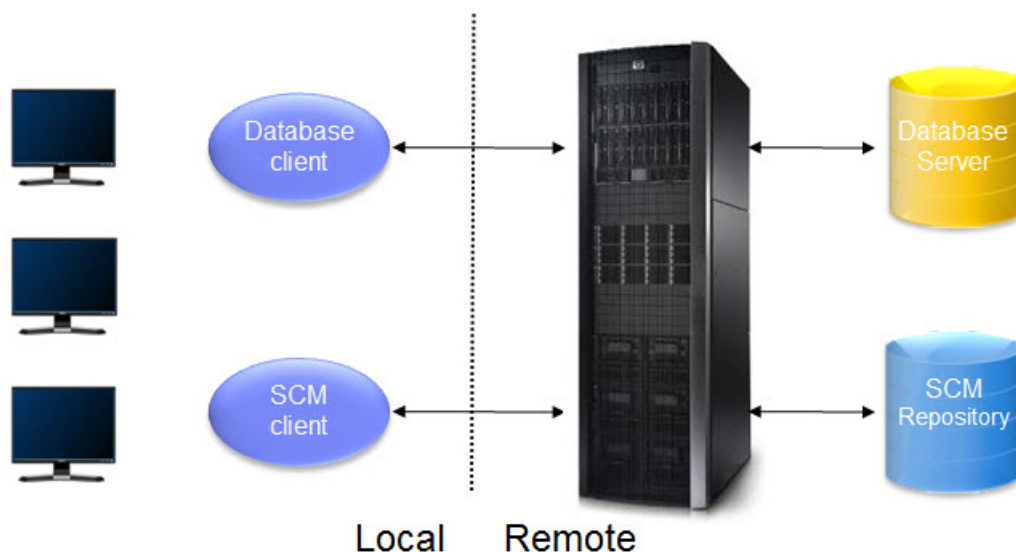


Figure 79: One or Multiple Users remote database

Remote Environments

In a remote environment, installation is done on the remote server and accessed by developers from their client machines.

- Genero applications are compiled, debugged, and executed on a remote server.
- Genero Studio Server must be installed on the remote server.
- Allows for the preservation of C function calls/libraries and/or system calls.
- Requires SSH access as well as Samba/NFS mounts.

The relational databases that the applications access can be stored on the same remote server as Genero Studio server, or on separate servers.

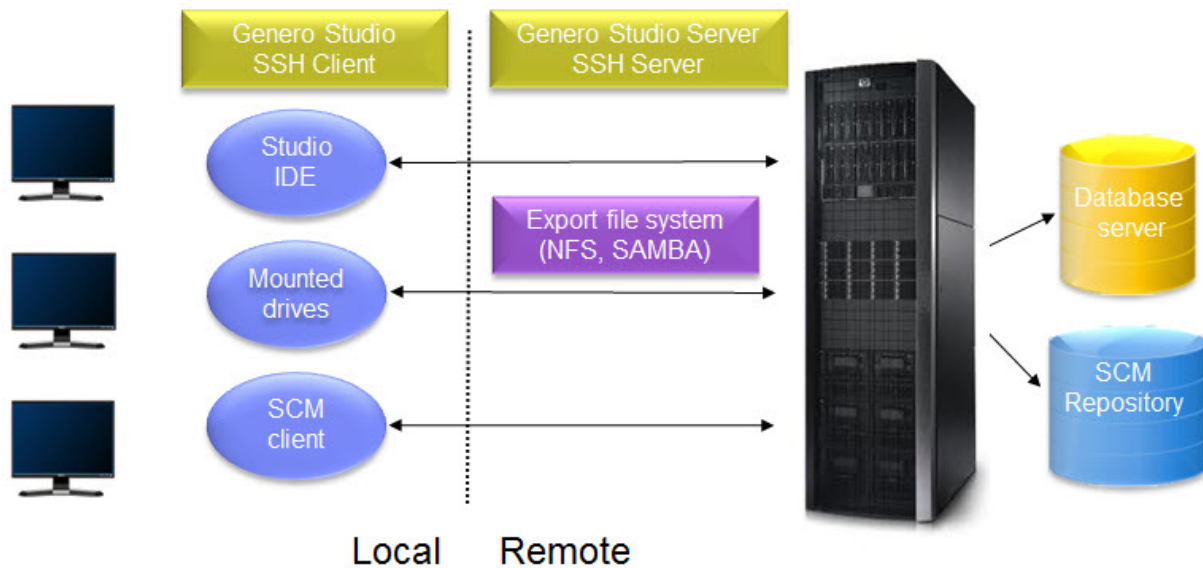


Figure 80: Remote Genero Studio environment

Default configuration

Configurations allow Genero Studio to locate the installed Genero components necessary to create, compile, build, and execute Genero applications on local or remote hosts.

Some default configurations are provided, based on the software in the distribution package installed in the default directories.

If you have installed your own Genero software, or have not installed Genero Studio in the default directories, you will have to define a Genero Studio configuration.

Even if you use the default configuration, you must set the [environment set](#) appropriate for your database client software, and add any missing variables or values. See [Access a database](#) on page 163.

Change the active configuration

When you have multiple configurations defined, only one of the configurations will be active.

The list in the bottom right corner of the main Genero Studio window displays the currently active server and client configuration.

To change the active configuration, select a server and/or display client configuration from the lists.

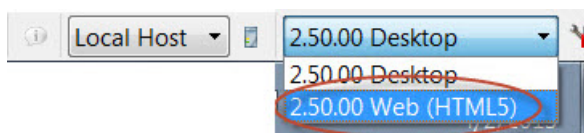


Figure 81: Change configuration

Alternatively, set the active configuration in the [Genero Configuration Management dialog](#) on page 171. The currently active configuration is in bold.

To change the active configuration, right-click on a configuration name and select **Set Active** from the contextual menu.

Configuring for BAM

When using the Business Application Modeler, you can specify both a global setup and a template-specific setup.

Factory setup

Factory setup is the standard installation.

Global setup

The global setup overrides the factory setup. It applies to all configurations for the current user for the Genero Studio installation on the current machine or remote machine.

Tools >> Global setup menu option allows you to customize the global setup.

Specific setup

A specific setup is a template-specific setup. It is for the current user, is for a template set used by the Business Application Modeler. Use is option to override the global setup for a specific set of templates. The [GSTSETUPDIR](#) on page 144 environment variable, set in the Genero configuration, defines the template set. If this environment variable is not set, or not specified for the current configuration, the specific setup options are disabled.

The **Tools >> Specific setup** menu option allows you to customize the specific setup.

Configure for Genero Mobile

Follow these instructions to complete the configuration for Genero Mobile for Android or iOS.

- [Configure Genero Mobile for Android](#) on page 118
- [Configure Genero Mobile for iOS](#) on page 127
- [Display to the Genero Mobile Development Client](#) on page 138

Configure Genero Mobile for Android

This configuration allows you to run a program from Genero Studio and display it to your Android physical or virtual device.

Requirements

- Java SDK. (Installed as part of [Install and configure Java SDK and Android SDK](#) on page 119)
- Android SDK. (Installed as part of [Install and configure Java SDK and Android SDK](#) on page 119)
- Either
 - An Android emulator (Android Virtual Device) (Created as part of [Display to an Android virtual device](#) on page 121)
 - An Android mobile device that runs a minimum of Android 4.0 (Ice Cream Sandwich) and is connected to the development machine via USB.
- Genero Mobile, which includes:
 - Genero Mobile for Android (GMA).
 - Genero Studio (launched when launching Genero Mobile) configured to communicate with the device or emulator.

Note: Genero Studio included with the Genero Mobile bundle is customized for mobile app development.

- You need an internet connection for the first time you build an Android package. During this first build, an automated process will download and install Gradle with all necessary extensions into a directory in your user directory. Gradle is a project automation tool, find out more about Gradle at <http://www.gradle.org>.

Install and configuration topics

- [Install and configure Java SDK and Android SDK](#) on page 119
- [Display to an Android virtual device](#) on page 121
- [Display to an Android physical device](#) on page 125
- [Configure multiple Android display devices](#) on page 126

Install and configure Java SDK and Android SDK

Follow this procedure to install and configure the Java SDK and Android SDK. Once configured, the **Tools >> Android Tools** menus are enabled in Genero Studio.

Before you begin:

- If a proxy is needed on your network, it must be defined in **Tools >> Preferences**.

You must configure for the Java and Android SDKs.

1. Install the Java Standard Edition Software Development Kit to a location of your choice: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. Make a note of the installation path.

Note:

- Genero Mobile requires Java SE Development Kit 7 (JDK 7) as a minimum.
- Choose the Java package (32 bit or 64 bit) that matches your Genero installation (32 bit or 64 bit).

2. Launch Genero Studio.
3. Select **Tools >> Genero Configurations** to open the Genero Configuration Management window.
4. Set the `JDK_HOME` variable to the location of the Java SDK you installed, for example `C:\Program Files\Java\jdk1.7.0`. To edit the `JDK_HOME` environment variable, highlight the **Java SDK** environment set and double-click on the `JDK_HOME` environment variable listed.

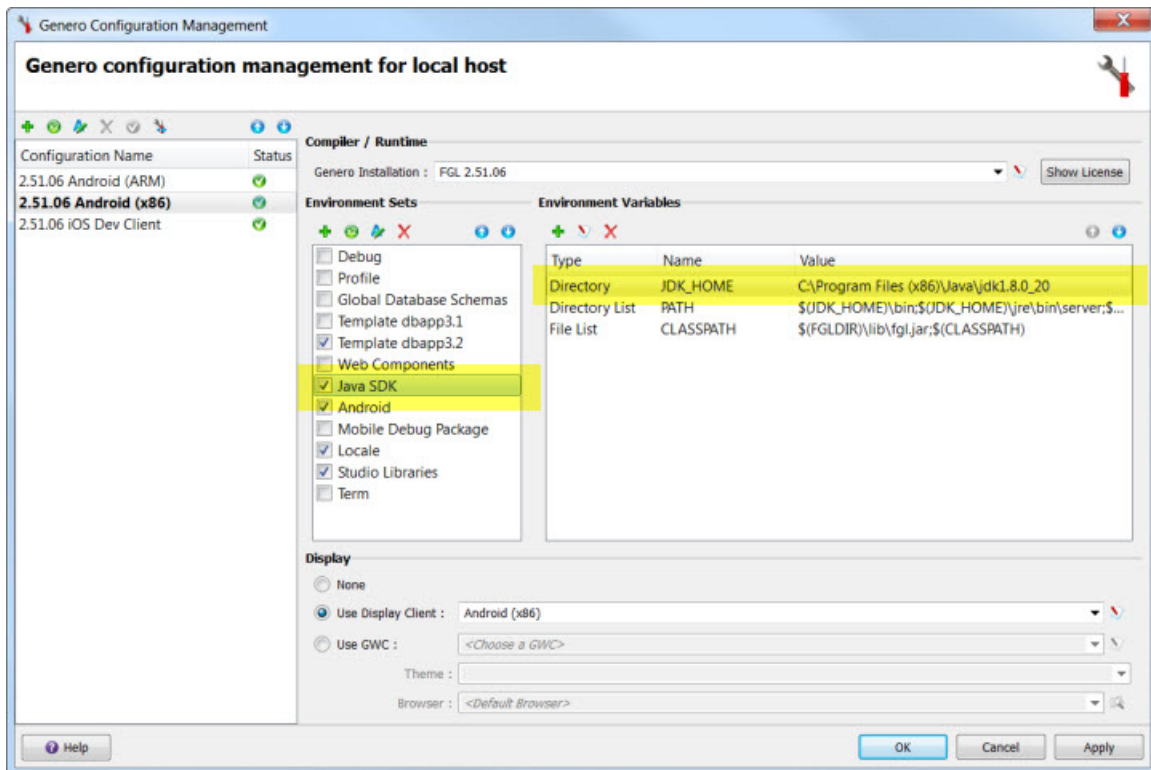


Figure 82: Set Java environment

- Download the Android Software Development Kit to a location of your choice. Go to this website, <http://developer.android.com/sdk/index.html>. For Windows or Linux install, select the **Download for Other Platforms** link. In the **SDK Tools Only** section, select the installer for your operating system. For Mac install, select **Use and Existing IDE**, and then **Download the SDK Tools for Mac**.
- Install the Android SDK to a location of your choice, but do not install in the Program Files directory. (If Android SDK has been installed in Program Files directory, Genero Mobile must be run as administrator to auto-configure Android SDK.)

Make a note of your installation path. If the Android SDK Manager launches, you may close it.

- Set the `ANDROID_HOME` variable to the location of the Android SDK you installed, for example `C:\Android\android-sdk`. To edit the `ANDROID_HOME` environment variable, highlight the **Android** environment set and double-click on the `ANDROID_HOME` environment variable in the list.

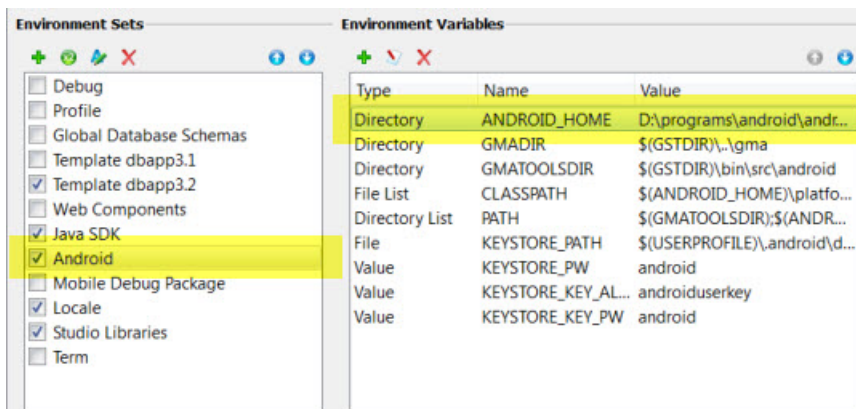


Figure 83: Set ANDROID_HOME in Android environment

- Right-click on your preferred Android configuration in the **Configuration Name** list and select **Set Genero Configuration Active**.
- Select **OK** to save the changes.

10. Select **Tools >> Android Tools >> Auto-configure Android SDK**. The updates will occur in a separate console window. Accept the license agreements as prompted.

Note: This step can take several minutes.

11. You are now ready to setup your Android physical or virtual device. See [Display to an Android physical device](#) on page 125 or [Display to an Android virtual device](#) on page 121.

You need an internet connection for the first time you build an Android package. During this first build, an automated process will download and install Gradle with all necessary extensions into a directory in your user directory. Gradle is a project automation tool, find out more about Gradle at <http://www.gradle.org>.

Configuration for extending Genero Mobile for Android

Follow this procedure if you plan to extend Genero Mobile for Android.

This procedure is only necessary if you plan to extend Genero Mobile for Android using Java. See *Extending the Language* in the *Genero Business Development Language User Guide*.

1. Create a copy of the GMADIR directory.

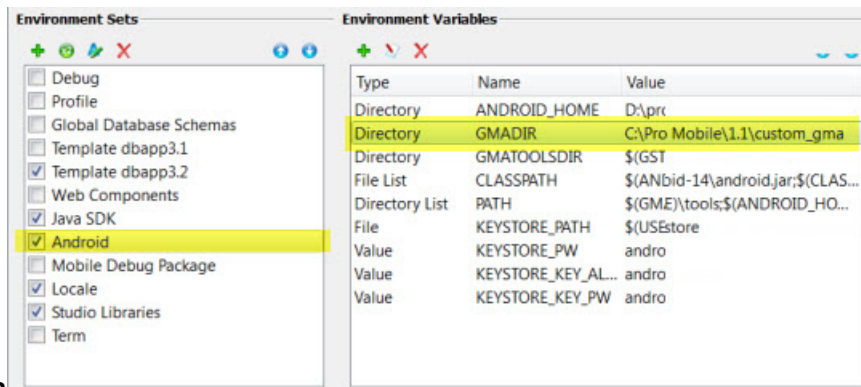
Find the default GMADIR at `GM_INSTALL_DIR/gma`. You should never update the default GMADIR, as it can be overwritten during an upgrade.

2. Update the GMADIR environment variable to point to your new copy.

a) Select **Tools >> Genero Configurations** to open the Genero Configuration Management window.

b) Set the GMADIR environment variable to the location of the copy of the GMA directory. To set the GMADIR environment variable, highlight the **Android** environment set and double-click on the GMADIR environment variable in the list.

Figure 84: Setting the



GMADIR

3. Follow the instructions in the *Genero Business Development Language User Guide* to extend Genero Mobile for Android.

Display to an Android virtual device

This configuration allows you to run a program from Genero Studio and display it to your Android virtual device (AVD).

Before you begin:

- Meet the requirements for Android emulator hardware acceleration. The Android emulator requires a processor with virtualization technology and a dedicated driver. Most recent Intel® processors support virtualization (VT-x, EM64T). See [Speeding Up the Android* Emulator on Intel Architecture](#).

1. [Install and configure Java SDK and Android SDK](#) on page 119.

2. Unplug any Android hardware devices connected via USB or [Configure multiple Android display devices](#) on page 126.

3. Select the Genero Configuration for **Android (x86)**.

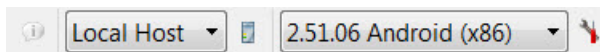


Figure 85: Genero Configurations list

You are now ready to create a virtual device and install Genero Mobile for Android (GMA) onto the virtual device.

Tip: Genero Mobile provides an Android Virtual Device (AVD), however you can elect to use another AVD, such as Genymotion.

Tip: If you create your own emulator, you must specify an external storage > 100 MB.

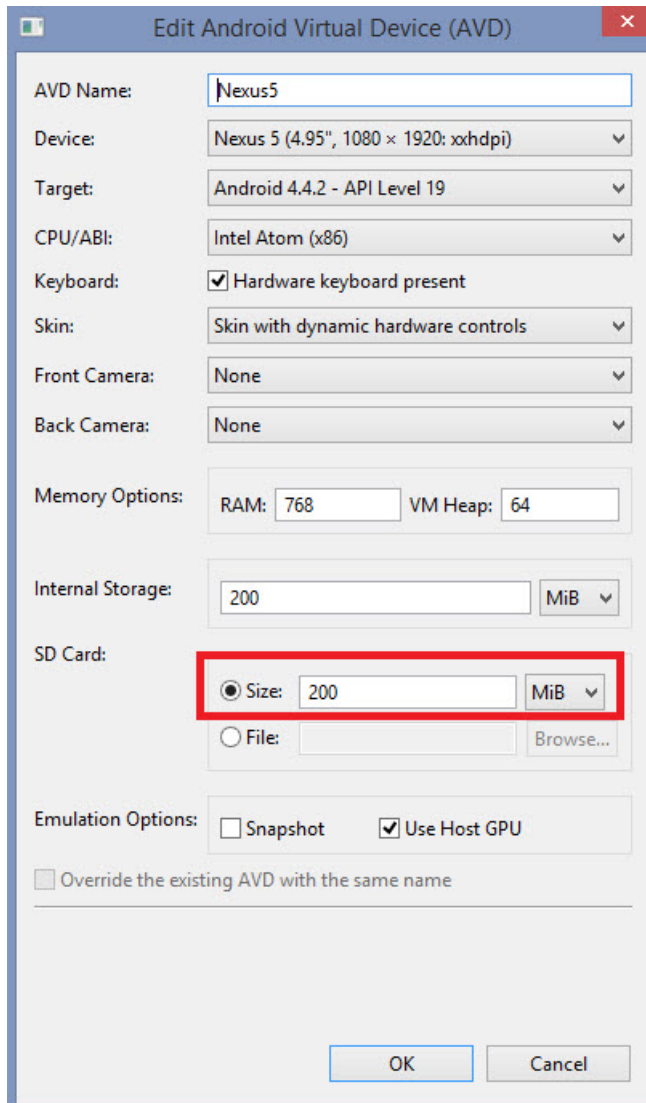


Figure 86: Android Virtual Device for Nexus5 with SD Card amount specified

4. Select **Tools >> Android Tools >> Create Android Virtual Device (x86)**. When the console appears, you will be asked if you wish to create a custom hardware profile. Press **Enter** to accept the default answer, no.
5. The Intel(R) Hardware Accelerated Execution Manager (HAXM) must be installed on the system. Go to your `ANDROID_HOME/extras/intel/Hardware_Accelerated_Execution_Manager` directory and double-click on the application to install it.

HAXM will improve x86 virtual device performance on Windows and MacOS X.

Warning: On MacOS 10.9, a Hotfix is needed (<http://software.intel.com/en-us/articles/intel-hardware-accelerated-execution-manager>).

6. Select **Tools >> Android Tools >> Launch Android Emulator (x86)** and wait for the emulator to finish loading. This can take a few minutes if you did not install HAXM. You will know it is finished loading when the emulator looks like a device screen.

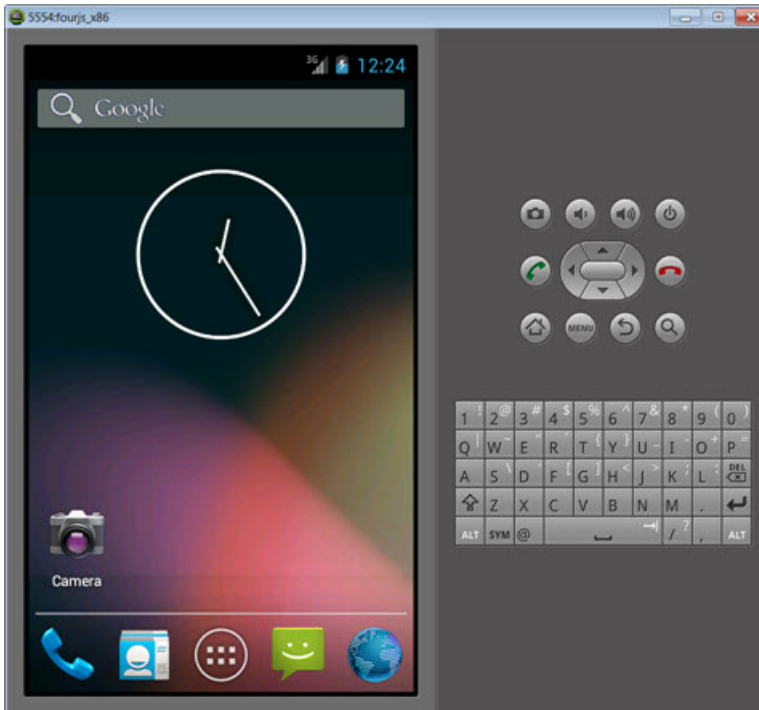


Figure 87: Running Android emulator

7. On the AVD, confirm that the AVD allows installation from unknown sources. Select **Settings >> Security** and confirm that **Unknown sources** is selected. Return to the Home screen and then go to the Apps screen.
8. From Genero Studio, select **Tools >> Android Tools >> Deploy Genero Mobile for Android**.
In the Genero Studio output panel you will see that the deploy started and finished. On the AVD you will see a new icon on the Apps screen labeled Genero Mobile.
9. Test your configuration. From Genero Studio, find the **OfficeStoreMobile** project and open it. Execute the **OrdersApp**. You should now see the Orders program running on the virtual device.
You can now run your own apps from Genero Studio to your virtual device. Your app will display to the virtual device currently running or you can [Configure multiple Android display devices](#) on page 126.

Troubleshoot Android SDK issues

Here are troubleshooting tips for issues you may encounter when using the Android SDK during the setup of your Genero Mobile development environment.

You follow the instructions to [Install and configure Java SDK and Android SDK](#) on page 119 and [Display to an Android virtual device](#) on page 121 and you get an error. What steps should you take?

1. Ensure that you have set the [ANDROID_HOME](#) environment variable correctly.
2. Run the auto-configure script for Android SDK (**Tools >> Android Tools >> Auto configure Android SDK**).

Troubleshoot Android emulator issues

Here are troubleshooting tips for issues you may encounter when using an Android Emulator during the development of your Genero Mobile app.

If you are having emulator issues, we recommend you read [Using the Emulator](#) on the Android Developers site.

My Android Emulator crashes intermittently

You followed the instructions to display to an Android virtual device, and you are getting intermittent crashes of the Android emulator when starting up. What is the issue?

It has been discovered that if you have an external monitor attached, or if you are moving your emulator between monitors in a dual monitor environment, it would save the location, which it would then use to restore when starting up. The location would then be read as an unexpected value, and would cause a core dump. To solve when this is the issue:

1. Navigate to the Android virtual device folder (.android/avd on a Mac, `userdir\.android\avd` on Windows).
2. Navigate to your Virtual Device (e.g., Nexus7.avd).
3. Edit `emulator-user.ini` and set:
 - `window.x = 0`
 - `window.y = 0`

If this is not the problem, then you may wish to try a different Android emulator, such as [Genymotion](#). Please check the system requirements before making the change!

To view the Android bug number for this issue: <https://code.google.com/p/android/issues/detail?id=40556>

My Android Emulator is too slow

Mobile emulators are virtual machines. As such, we highly recommend a desktop or laptop with CPU virtualization support and sufficient memory. If you do not have this, you will find the mobile emulators will run very slowly.

Important: The following system requirements are recommendations based on user experiences.

For Mac users:

- Mac or Macbook Pro (Intel based)
- OS X 10.7 or higher
- 8+ Gb RAM
- VT-x enabled (may require firmware upgrade, see <http://support.apple.com/kb/TS2744>)

For Windows users:

- Windows 7 or higher
- 8+ Gb RAM
- Intel CPU with VT-x or AMD-v CPU
- VT-x or AMD-v enabled (may require BIOS configuration)
- You can check if your CPU has VT-x with this Intel CPU identification tool: <http://www.intel.com/support/processors/tools/piu/sb/CS-014921.htm>

At this time, we have no statistics regarding system requirements for Linux users.

Display to an Android physical device

This configuration allows you to run a program from Genero Studio and display it to your Android device.

Note: These configuration instructions are for a Google Nexus 7, the default platform used, and are provided as an example. Check the device manufacturer documentation or web site for information related to your specific device.

1. [Install and configure Java SDK and Android SDK](#) on page 119.
2. Close the Android emulator if it is currently open or [Configure multiple Android display devices](#) on page 126.
3. Select the Genero Configuration for **Android Device (ARM)**.

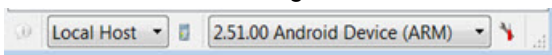


Figure 88:

You are now ready to install Genero Mobile for Android (GMA) onto your device.

4. Connect your Android device to your computer's USB port.
5. To deploy and validate your app to a device, you must configure the device for development. Enable USB debugging on your device by checking the **Allow USB debugging** option. Select **Always allow from this computer** so that you do not have to repeat this step.

On Android 4.0 and newer, it's in **Settings >> Developer options**. On Android 4.2 and newer, **Developer options** is hidden by default. To make it available, go to **Settings >> About phone** (or **About tablet**) and tap Build number seven times. Return to the previous screen to find Developer options and to allow **USB Debugging**.

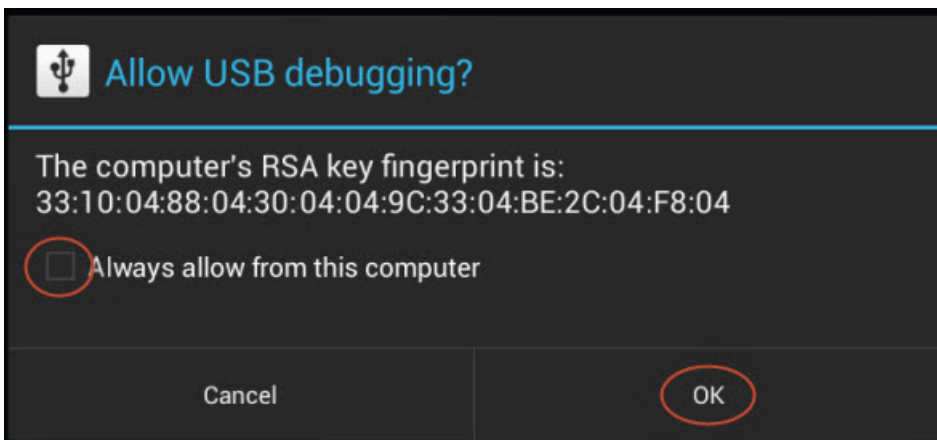


Figure 89:

6. Go to **Settings >> Storage >> USB computer connection**. (This option is in the top right menu.) To see the internal storage, set the USB computer connection mode to "Media device (MTP)".
7. A USB driver update will be required (Windows). Follow these instructions to find and install the appropriate USB driver for your device: <http://developer.android.com/tools/extras/oem-usb.html#InstallingDriver>.
8. Set your device to allow installation from unknown sources. Select **Settings >> Security** and tap **Unknown sources** to select it. Return to the Home screen and then go to the Apps screen.

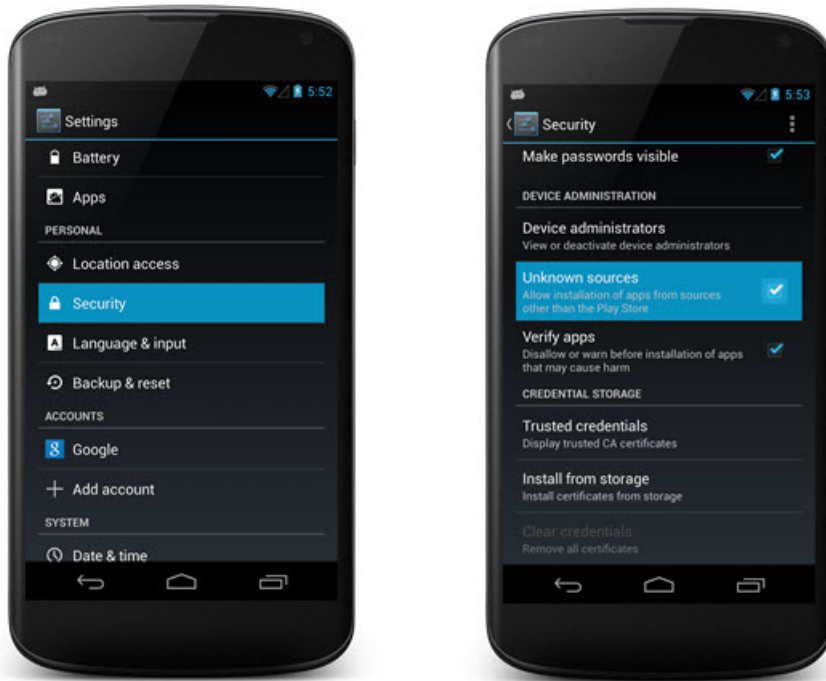


Figure 90: Install from unknown sources

9. Check the communication from the computer to your device. From Genero Studio, select **Tools >> Android Tools >> List Devices**. In the output panel you should see a list of devices attached. If several devices are connected, you will need to define device ID in the display client configuration. If a device is offline, restart the state **Allow USB debugging**.
10. From Genero Studio, select **Tools >> Android Tools >> Deploy Genero Mobile for Android**.
In the output panel you will see that the deploy started and finished. On the device you will see a new icon on the Apps screen labeled Genero Mobile. This is the development client.
11. Disable installation from unknown sources on your device. Select **Settings >> Security** and tap **Unknown Sources** to deselect it.
12. Test your configuration. From Genero Studio, find the **OfficeStoreMobile** project and open it. Execute the **OrdersApp**. You should now see the Orders program running on your device.
You can now run your own apps from Genero Studio to your device. You can also deploy an app to the device. See [Deploy a mobile app for testing](#) on page 998 for details.

Configure multiple Android display devices

Use this procedure to configure multiple Android display devices, for example a phone and a tablet and an emulator.

By default, the **device ID** is not set in the **Display client** configuration and Genero Studio uses the first connected device it finds. If you want to connect multiple devices at the same time, you can create multiple configurations, each one specific to a **device ID**.

1. Select **Tools >> Android Tools >> List Devices**. The list of connected devices appears in the output along with their respective device IDs.

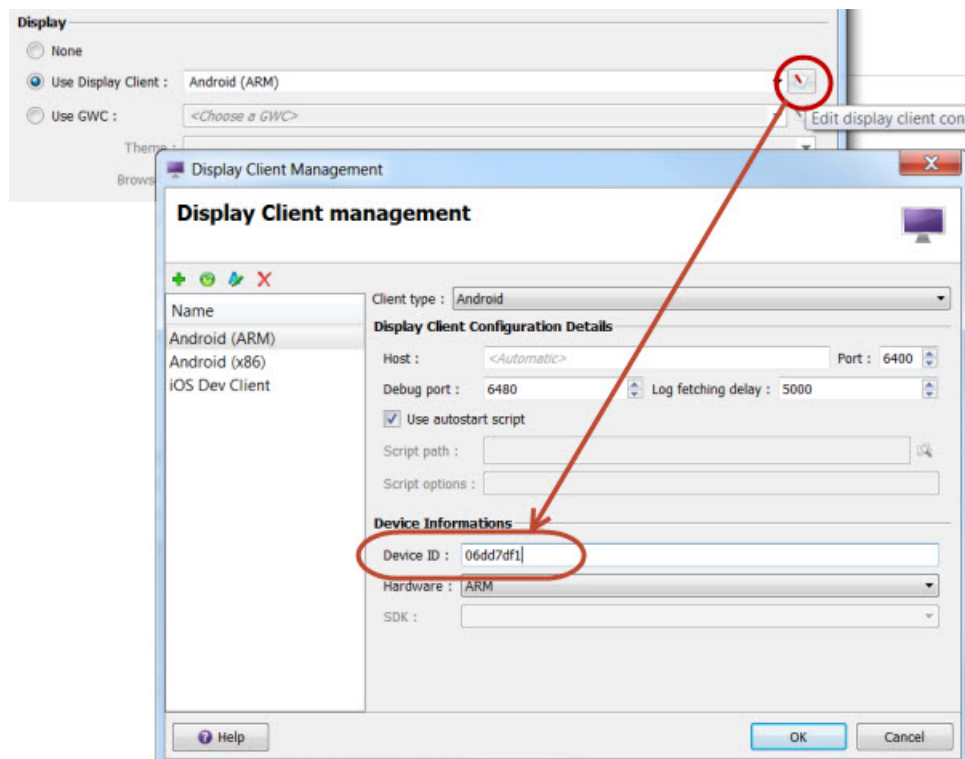
```
List of devices attached
06dd7df1 device
emulator-5554 device
```

- Copy the device ID of your device. Android virtual devices are considered a device by this utility and will appear in the list as well. If you have multiple devices listed and you are not sure which one is the right one, disconnect all of them except the one you wish to configure.
- In the Display Client management dialog, enter the ID into the **Device ID** field. You may edit the default configuration or add new Display Client configurations.

We recommend having a different port for each client available on the client machine. For example:

- 6500 for an Android Nexus 7 device
- 6501 for an Android Nexus 5 device
- 6502 for an Android Emulator
- 6503 for an iOS simulator

and so on.



- If you have more than one USB device connected, create a separate Genero configuration for each Android device you have connected. You can now start the emulator and connect all the devices you configured via USB. Selecting a Genero Configuration will select the device (or emulator) to use.

Configure Genero Mobile for iOS

Requirements for configuring Genero Mobile to develop apps for iOS.

Requirements

This configuration allows you to run a program and display it to your iOS physical or virtual device.

- Genero Mobile, which includes Genero Studio customized for mobile app development
- A Mac running a minimum of Mac OS 10.8. See [Install and configure Xcode](#) on page 128 for version details.
- Xcode. See [Install and configure Xcode](#) on page 128.
- Either the iOS Simulator or an iOS physical device connected to machine via USB
- A standard, free Apple ID account or a purchased developer account

Note: A developer account is not required for creating apps that run on the simulator. You will need an account once you want to:

- Install an app on an iOS device (even your own one for debugging)
- Distribute an app on the iOS app store
- Distribute in-house iOS apps to your employees
- Genero Studio (launched when launching Genero Mobile) configured to communicate with the device or simulator

Install and configuration topics

- [Install and configure Xcode](#) on page 128
- [Display to an iOS simulator](#) on page 128
- [Display to an iOS physical device](#) on page 129
- [Configure multiple iOS display devices](#) on page 137
- [Display to the Genero Mobile Development Client](#) on page 138

Install and configure Xcode

Follow this procedure to configure for iOS. Once configured, the **Tools >> iOS Tools** menus are enabled in Genero Studio.

1. Download and install Xcode from the Mac app store.
2. Install the Xcode command-line tools. They can be installed from Xcode by selecting **Xcode >> Open Developer Tool >> More Developer Tools**.
3. Launch Genero Studio.
4. Select **Tools >> Genero Configurations** to open the Genero Configuration Management window.
5. You are now ready to [Display to an iOS simulator](#) on page 128 or [Display to an iOS physical device](#) on page 129.

Configuration for extending Genero Mobile for iOS

Follow this procedure if you plan to extend Genero Mobile for iOS (GMI).

This procedure is only necessary if you plan to extend Genero Mobile for iOS using Objective-C.

1. Create a copy of the `GMIDIR` directory.
Find the default `GMIDIR` at `GM_INSTALL_DIR/gmi`. You should never update the default `GMIDIR`, as it can be overwritten during an upgrade.
2. Update the `GMIDIR` environment variable to point to your new copy.
 - a) Select **Tools >> Genero Configurations** to open the Genero Configuration Management window.
 - b) Set the `GMIDIR` environment variable to the location of the copy of the GMI directory. To set the `GMIDIR` environment variable, highlight the **iOS** environment set and double-click on the `GMIDIR` environment variable in the list.
3. Follow the instructions in the *Genero Business Development Language User Guide* to extend Genero Mobile for iOS.

Display to an iOS simulator

This configuration allows you to run a program from Genero Studio and display it to your iOS Simulator. This configuration *does not* require an Apple iOS developer account.

Before you begin:

- [Install and configure Xcode](#) on page 128.
1. Launch Genero Studio.
 2. Confirm that the Genero Configuration for **iOS Simulator** is selected.

Note: The Device ID field in iOS Simulator is used to specify which simulator to use when deploying application or Genero Mobile Client, or launching Genero Mobile Client when executing or debugging an application.

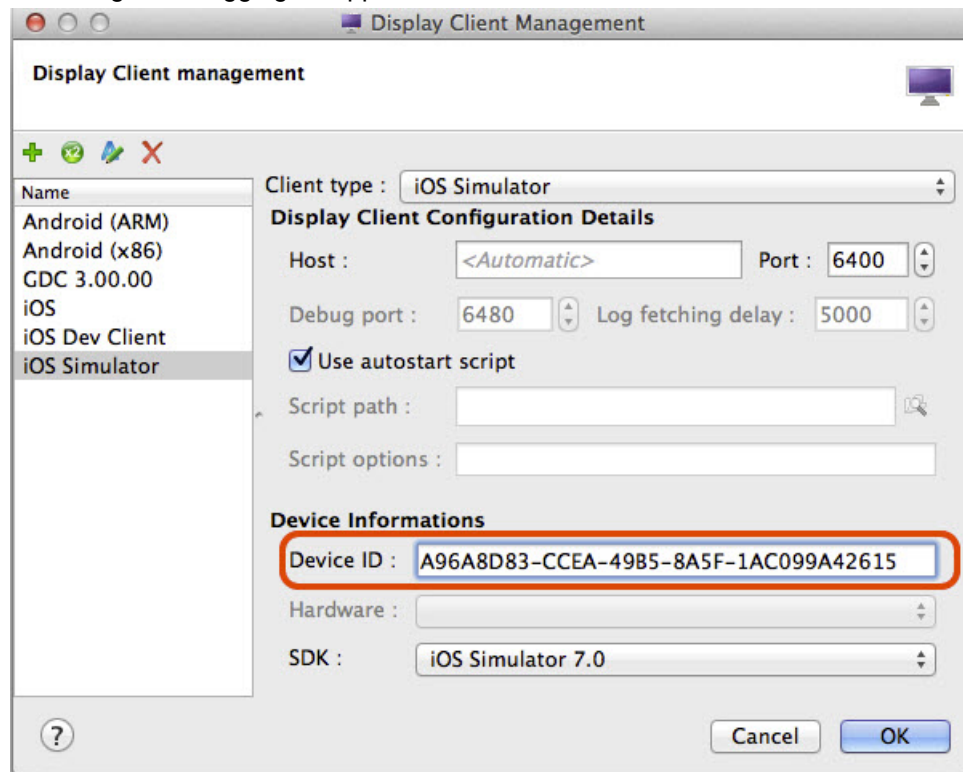


Figure 91: Display Client Management dialog with Device ID field

You can see the list of simulator identifier by calling **Tools >> iOS Tools >> List Devices** action. You access the **Device ID** field by opening **Genero Configuration Management** dialog then opening the list of display clients and selecting an iOS Simulator display client. The value you can put in this field should be a substring of a simulator as displayed by **List Devices** action.

It can be the internal identifier or the label or a part of it. If the field is not filled, then "iPhone 4S" simulator will be used, as it is the less resource-intensive one.

3. Select **Tools >> iOS Tools >> Launch iOS Simulator**. The simulator should appear. Change the simulator hardware to match the device you want to simulate, such as an iPhone.
4. Select **Tools >> iOS Tools >> Deploy Genero Mobile for iOS**.

In the Genero Studio output panel you will see that the deploy started and finished. On the Simulator you will see a demo app running.

5. Test your configuration. From Genero Studio, find the **OfficeStoreMobile** project and open it. Execute the **OrdersApp**. You should now see the Orders program running on the simulator.
6. You can now run your own apps from Genero Studio to the simulator.

Display to an iOS physical device

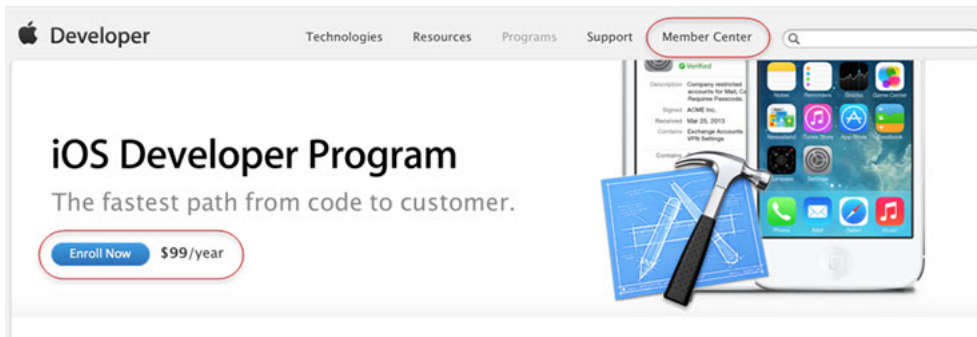
This configuration allows you to run a program from Genero Studio and display it to your iOS device. This configuration *does* require an Apple iOS developer account.

Before you begin:

- [Install and configure Xcode](#) on page 128.

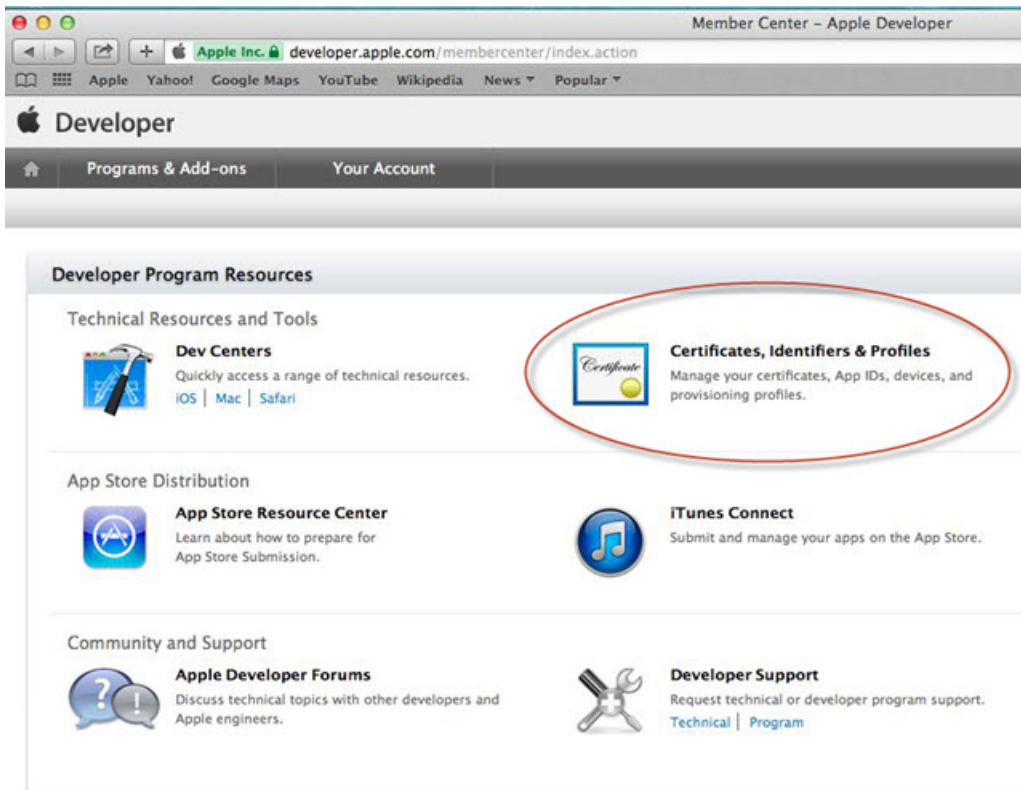
1. Log into the iOS Developer Center. If you are a member of the iOS Developer Center then just log in. If not, follow these steps to enroll in an iOS developer program. Please note, it can take Apple a day or two to approve memberships.

- a) Go to <http://developer.apple.com> and select the **iOS Developer Program** option.
- b) Select the **Enroll Now** option and follow the instructions. You will need to enroll in either a Developer or Enterprise program. When you are finished you will have an Apple ID with which to log in. If you are a member of the iOS Developer Program select **Member Center** to log in.



2. Generate and import a development certificate. The Development Certificate will be used to allow you to view your program on your iOS device which can be an iPhone, iPod, or iPad.

- a) Select **Certificates, Identifiers & Profiles**.



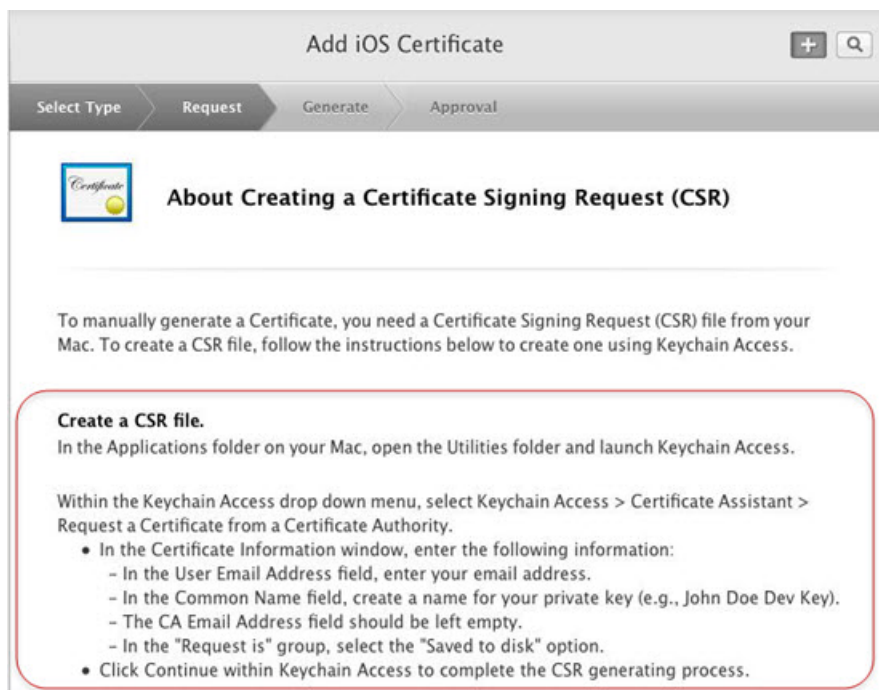
- b) Select **Certificates** under the iOS Apps section.



- c) Under **Certificates**, select **Development** followed by the **+** symbol.



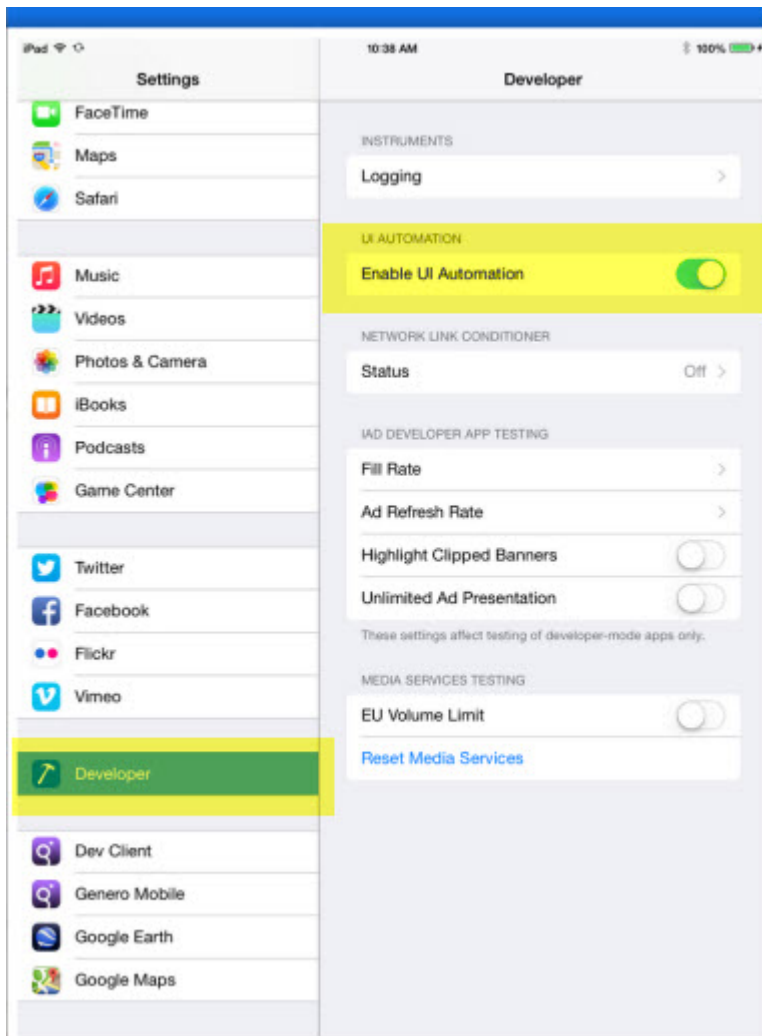
- d) Follow the instructions, on the **Select Type** page and select the **iOS App Development** option for an **iOS App Development** certificate then click **Continue**.
- e) Follow the instructions on the page to create a **CSR file** then click **Continue**.



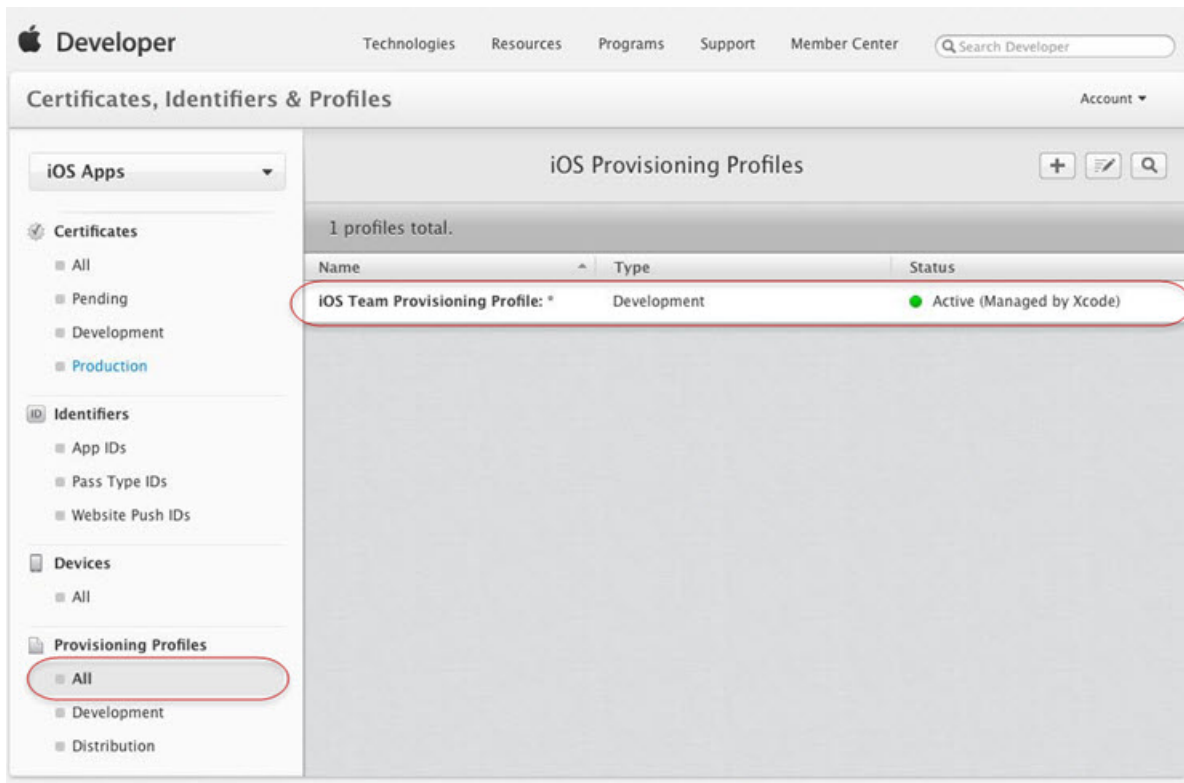
- f) Your certificate request is now available. Now you can go back to the Development Certificate section still active in your browser and click **Choose file...**
- g) Navigate to the file you just saved and choose that file.
- h) Click **Generate**.
- i) Once the certificate is generated, click **Download**. The certificate will download into your Downloads folder.
- j) Double-click this file to install it into Keychain.
- k) When done, your new certificate should be listed in the **Certificates** list.



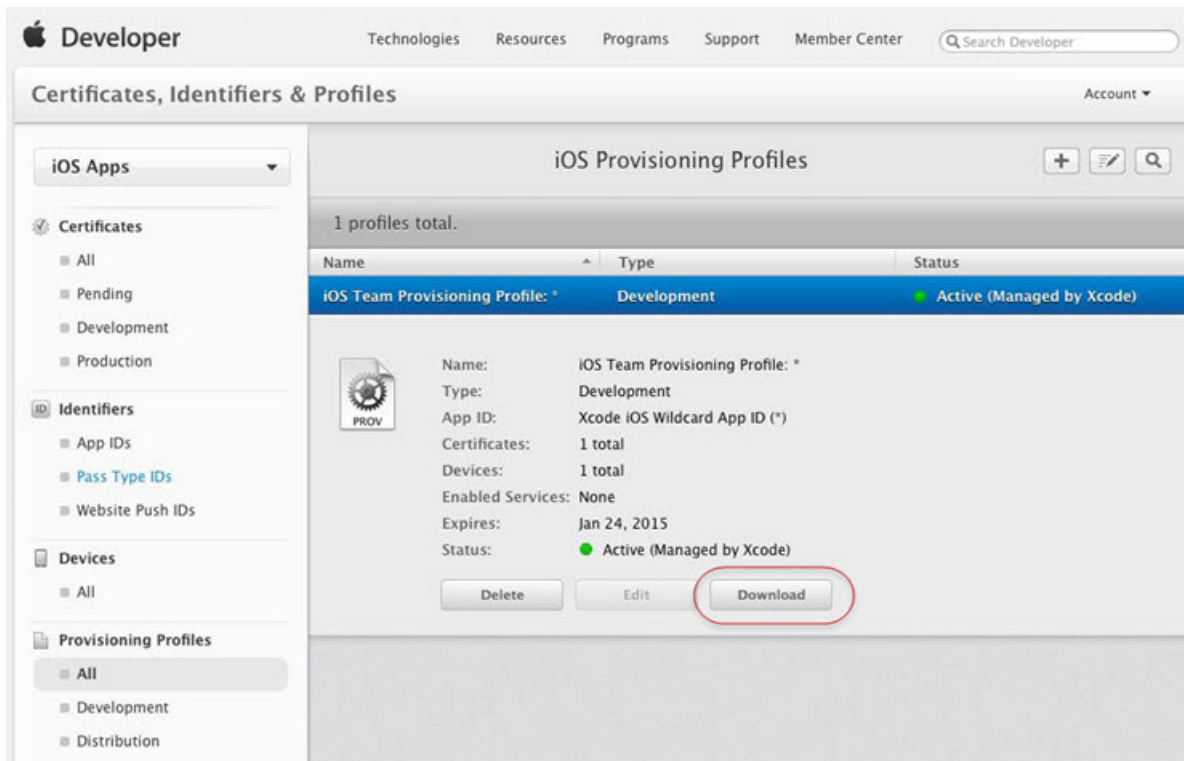
3. Provision your device for development. If you have only an iOS developer account, you need to register each device you will use to test your iOS app. These steps are *not* necessary if you have an iOS enterprise account. Follow Apple's instructions for [provisioning your device](#).
4. For iOS 8 and above, enable UI automation.
 - a) Open the Settings app.
 - b) Select Developer.
 - c) Turn on Enable UI automation.



5. Go back to the Member Center and you should have a provisioning profile granted. Please note, it can take some time before Apple changes the status from Pending to Active.

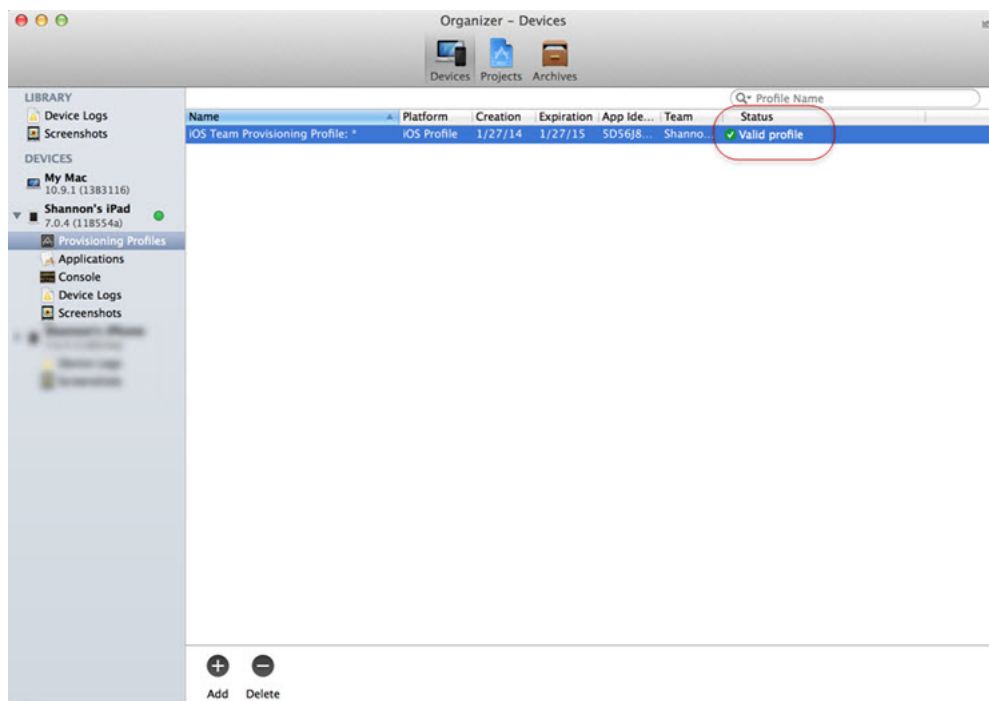


6. Select the Provisioning Profile and click **Download**.



7. From the Xcode Organizer (**Xcode >> Window >> Organizer**), select Devices.

8. Drag the provisioning profile you downloaded into the Xcode Organizer window. You should see a valid provisioning profile installed with a green check mark.



9. Launch Genero Studio.

10. Select **Tools >> Genero Configurations**.

11. Select the **iOS** environment set.

12. Set the `IDENTITY` environment variable in the **iOS** environment set, to identify which certificate to use from those defined in Keychain Access.

The certificate name generally contain the first and last name of the developer as defined in your Apple ID account. You can find it at <http://appleid.apple.com/>.

To view the list of certificates in the Keychain Access, leave Genero Studio and complete these steps:

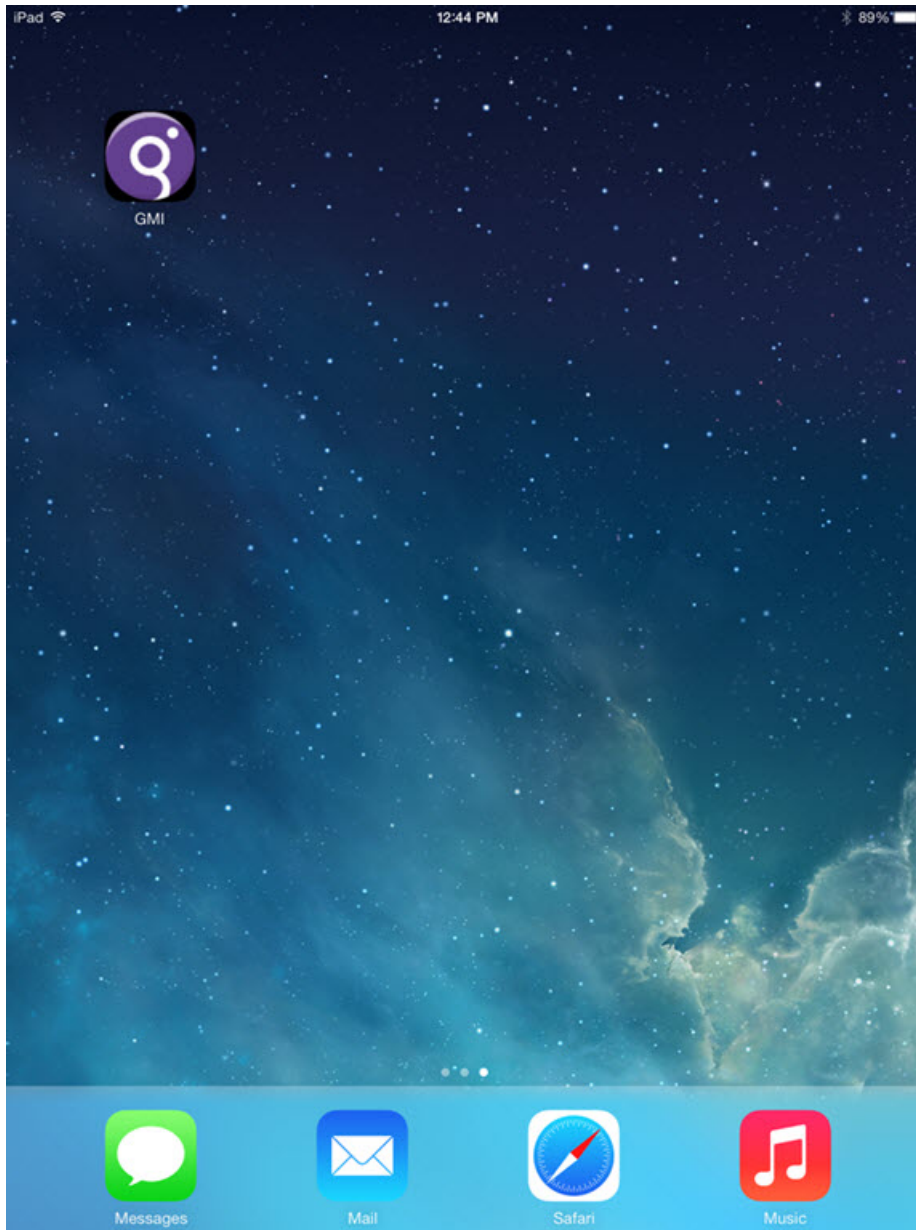
- a) Open **Applications >> Utilities >> Keychain Access**.
- b) Select **login** in the **Keychain** section.
- c) Select **My certificates** in the **Categories** section.

The list of certificates displays. Use this list to find the sub string to enter for the `IDENTITY` environment variable. At a minimum, the `IDENTITY` environment variable must contain just enough characters to identify the certificate (amongst those listed in Keychain Access) used to sign the package. In theory, it can be as small as two letters, if those two letters are sufficient enough to identify the certificate.

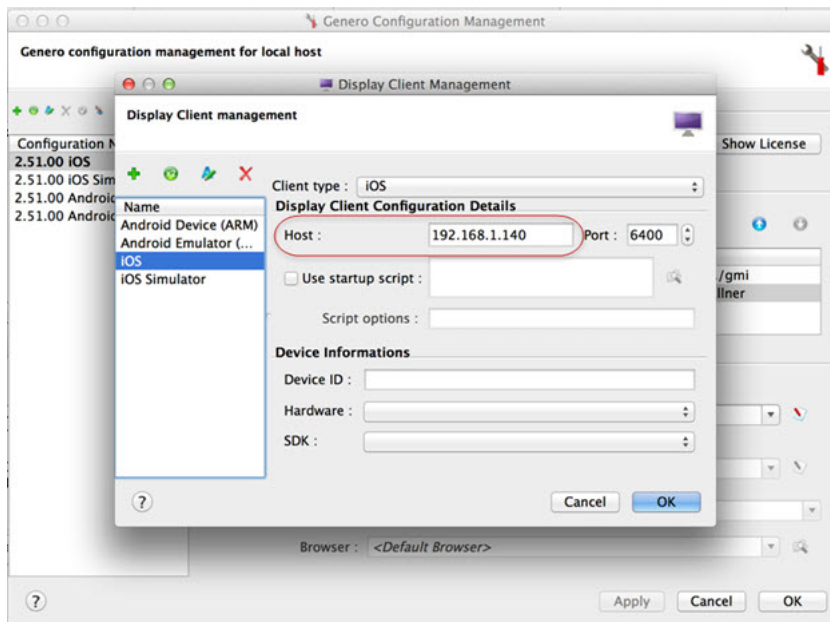
13. Set the `PROVISIONING_PROFILE` environment variable to the provisioning profile file you downloaded.

14. Select **Tools >> iOS Tools >> Deploy Genero Mobile for iOS**. A USB connection is required to deploy to the device.

In the Genero Studio output panel you will see that the deploy started and finished. On the device you will see a new icon labeled GMI.



15. Set up your device as a display client. Wi-Fi is used to display the app to the device in developer mode.
- Confirm that your computer and your mobile device are on the same Wi-Fi network.
 - Get the IP address of your device. (From your device, select **Settings >> Wi-Fi** and select the Wi-Fi network to see the network details and IP Address.)
 - Edit the **iOS** Display client setting for the **iOS** Genero configuration. In the **Host** field, enter your device's IP address.
 - Select OK to save the changes.



16. On the device, tap the GMI app to launch it.

17. Test your configuration. From Genero Studio, find the **OfficeStoreMobile** project and open it. Execute the **OrdersApp**. You should now see the Orders program running on the device.

18. You can now run your own apps from Genero Studio to the device.

Configure multiple iOS display devices

Use this procedure to configure multiple iOS display devices, for example a phone and a tablet.

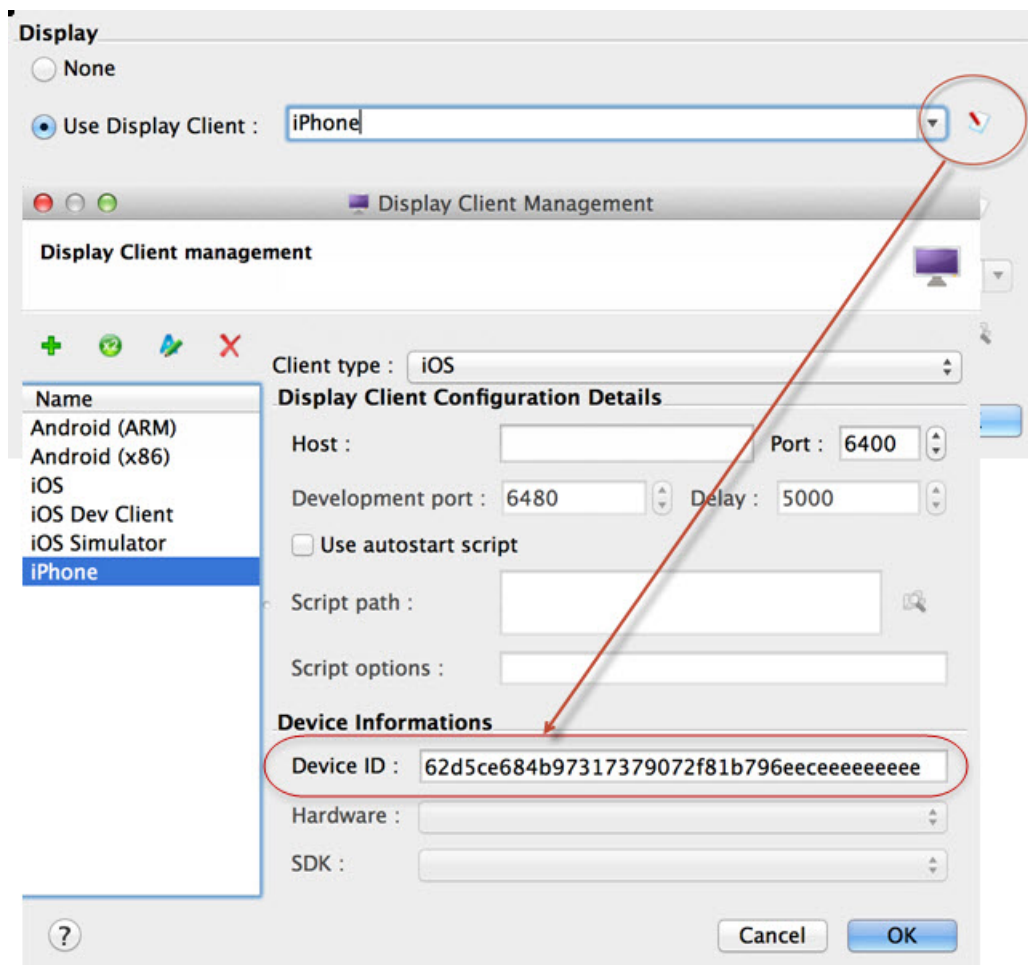
By default, the **device ID** is not set in the **Display client** configuration and Genero Studio uses the first connected device it finds. If you want to connect multiple devices at the same time, you can create multiple configurations, each one specific to a **device ID**.

1. Select **Tools >> iOS Tools >> List Devices**. The list of connected devices appears in the output along with their respective device IDs.

```
List of iOS devices
iPad: 93f794ca3b3faae092980aba8410102ffffffffffff
iPhone: 62d5ce684b97317379072f81b796ecccccccc
```

2. Copy the device ID of a device.

3. In the Display Client management dialog, enter the ID into the **Device ID** field. You may edit the default configuration or add new Display Client configurations.



- If you have more than one USB device connected, create a separate Genero configuration for each device.
Selecting a Genero Configuration will select the device to use.

Display to the Genero Mobile Development Client

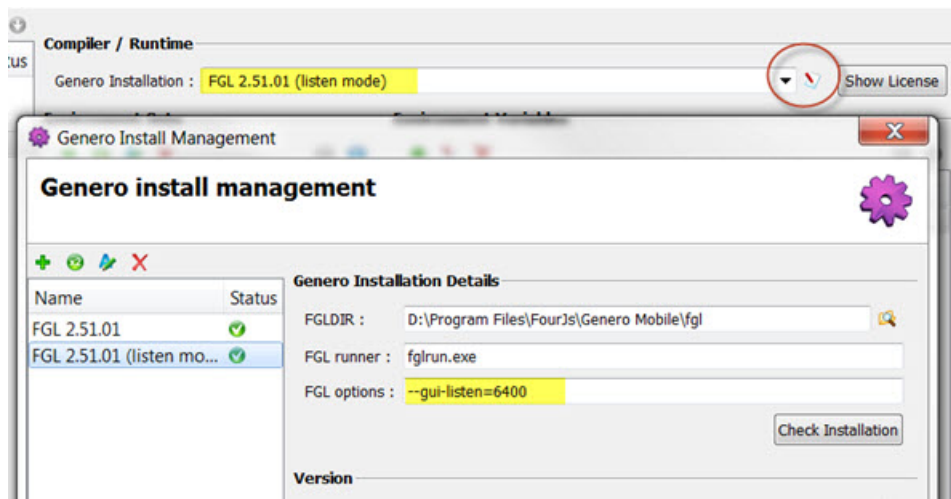
Available from Apple's App Store, the Genero Mobile Development Client serves two purposes: It allows you to view your app on an iOS device regardless of your development machine operating system, and it allows you to run a demo app.

Before you begin:

- The mobile device and the development machine must be connected to the same wireless network.
- The Genero Mobile Development Client app (available from the App Store) must be installed on the mobile device. To locate the Genero Mobile Development Client in the App Store, simply search on the full name of the app. After installation, the app appears with the name **Dev Client** and the standard Genero icon.

- Launch Genero Studio.
- Select the **iOS Dev Client** configuration.

The iOS Dev Client is configured for the DVM (fglrn) to listen on port 6400. To listen on a different port, modify the **Genero Installation FGL options** setting to a different port.



3. Run your app.

- Execute your app from Genero Studio. If you have a firewall, you must allow the client to communicate on your network.
- Alternatively, start the app on the server from the command line using the `--gui-listen` option. For example:

```
% fglrun --gui-listen=6400 helloworld.42m
```

The app starts and waits for the Genero Mobile Development Client to connect.

4. From your device, start the **Dev Client** app.

5. In the **URL** field, enter the following URL.

```
fgl://yourdesktop-ip:port-number
```

Replace *yourdesktop-ip* with the IP address of your development machine. Replace *port-number* with the port number specified in the iOS Dev Client configuration. The default is 6400. The firewall on your desktop needs to be open for the port used. For example:

```
fgl://192.168.0.101:6400
```

6. Set the **Timeout** and **Permanent retry** options as needed. For the **Timeout** option, use the spinner to set the number of seconds for the timeout. Setting the timeout to zero indicates that you are not setting a time limit. If you have no service waiting, it will immediately fail. Turn **Permanent retry** on or off. If you switch on, and kill the app, then it tries for some seconds to connect, then retries and keeps retrying.
7. Tap **Connect**.
Either the app will display, or you will get an error message in the **RESULTS** area of the interface. When you exit the app, you are returned to the developer interface.

Setting up a local environment

These topics assist you in setting up a local Genero Studio environment.

- [Local environment software requirements](#) on page 140
- [Define local Genero installations, GAS configurations, and environment sets](#) on page 140

Local environment software requirements

Install the complete Genero stack on the local machine when a developer will work independently and compile files locally.

A local environment requires the following components:

- Genero Studio
- Installed Genero Business Development Language (BDL) compiler and runner (DVM). See [Compiler / Runtime configuration \(Genero Installations\)](#) on page 172.
- Installed display client such as GDC, GWC, GMI, GMA depending on your application development (desktop, web, mobile). See [Display clients](#) on page 146.

If the project requires a database, the database (Informix®, Oracle, etc.) and its accompanying database client (CSDK, Oracle client, etc.) can be installed on the local machine or configured for remote access:

- The database server installed on a remote host
- The database client installed on the local machine

Project and source files must be local, but can be shared using a version control system if a networked drive (or samba) is configured.

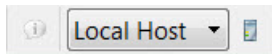
Each database environment has its own client character set configuration. You must configure the database client locale properly in order to send/receive data to the database server. The settings for database client locale and application locale must match. See *Localization: Database Client Settings* in the *Genero Business Development Language User Guide*.

See the installation documentation and release notes for additional information.

Define local Genero installations, GAS configurations, and environment sets

Local default configurations based on your installation are set up for you. Modify the configurations if the defaults are not set up for your needs.

1. Select the local host from the list in the lower right corner.



2. Select **Tools >> Genero Configurations** to display the **Genero Configuration Management** dialog for the local host.
3. Add new configurations as needed to add [Compiler / Runtime configuration \(Genero Installations\)](#) on page 172, [Environment sets](#) on page 140, and [Web: GAS/GWC configurations](#) on page 147.
4. Save the changes.

Environment sets

Define sets of environment variables to add to or overwrite the default environment. These environment settings are also usable by each Genero Studio client that adds a remote Genero Studio Server to its list.

Select **Tools >> Genero Configurations** to display the configurations for the currently selected server. Add and modify environment sets as needed.

When you launch any Genero Studio sub-process (run, GDC, compilation, debug, and so on), Genero Studio sets all redefined environment variables according to the current context (the Projects context may differ from the File Browser context).

Select the name of an environment set to view/modify the Environment Set variables. Right-click in the **Environment Set** area and select **Add Environment Set...** to add a new set of environment variables.

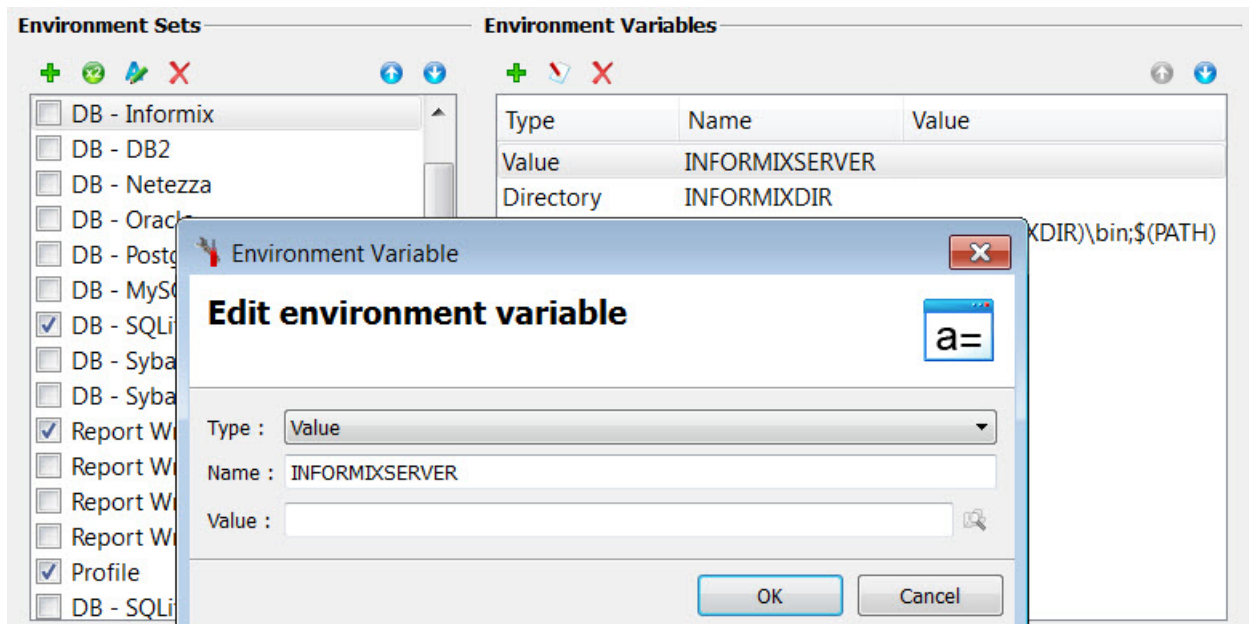


Figure 92: Environment sets

See [Add or edit environment variables](#) on page 144 for more details on using the Environment Variable dialog.

Default environment sets

Some default environment sets are included.

Debug

Contains variables that set the [debug level](#) (*FGLSQLDEBUG* for FGL/SQL, *FGLWSDEBUG* for Web Services, and *GREDEBUG* for Report Engine.) The values are already set to 9, the highest debug level, but can be changed to any number between 1 and 9. Debug files will be generated in the current directory.

DB - xxxx

Contains some variables required by database clients from the various vendors. The values for the variables must be entered before using these environment sets. See [Database server/user information](#) on page 308 for information about the specific variables for each database.

Important: This feature is not supported on mobile platforms.

Global Database Schemas

Defines the schema files (*4db*, *4dbx*, *sch*) to be loaded by default into the Meta-schema Manager when Genero Studio is launched. Includes setting for [GSTSCHAMANAMES](#) and *FGLDBPATH*.

Java SDK

Environment settings for the installed Java SDK. Adds the Java executables path to the *PATH* environment variable. See the *Genero Mobile User Guide* for more information.

Profile

Contains the environment variable *FGLPROFILE*; for the value, enter the path of the current

	<p>configuration (<code>FGLPROFILE</code>) file to be used by the system before enabling this environment set; the default location is <code>FGLDIR/etc/fglprofile</code>. If you are using a remote environment, the value should be set to the location on the remote host. See Access a database on page 163.</p>
Report Writer	<p>Contains the environment variables set by default to the installation directory of Genero Report Engine. This set is automatically enabled, as the environment variables are required when working with reports.</p> <p>Important: This feature is not supported on mobile platforms.</p>
Templates	<p>Contains environment variables set by default to the template directory being used for Business Application Modeling. Includes setting for <code>GSTSETUPDIR</code>. See The code generation template set on page 942.</p>
Web Components	<p>Defines <code>GSTWCDIR</code>, the directory in which the WebComponent XML files and optional icon files are stored.</p>
Android	<p>For Genero Mobile, environment settings for the installed Android SDK. See the <i>Genero Mobile User Guide</i> for more information.</p>
iOS	<p>For Genero Mobile, environment settings for iOS. See the <i>Genero Mobile Developer Guide</i> for more information.</p>
Mobile Debug Packages	<p>For Genero Mobile, sets the <code>DEBUG_PACKAGE</code> environment variable to <code>TRUE</code> so that packages built will include options for debugging. See <i>Debugging a mobile app</i> in the <i>Genero Mobile Developer Guide</i>.</p>
Locale	<p>Sets the <code>LANG</code> environment variable. See Language support (text encoding) on page 163 and <i>Language and character set settings</i> in the <i>Genero Business Development Language User Guide</i> for more information.</p>
Studio Libraries	<p>Contains the <code>GSTLIBRARYDIR</code> environment variable which specifies the location of the libraries used with the database generation script.</p>
Term	<p>Contains the <code>GSTTERM</code> (Linux, Windows) environment variable which specifies how to open a terminal on the client machine (Genero Workspace Window, for example). For some Linux operating systems which do not have <code>xterm</code>, use this to specify the terminal name and the arguments to launch it. The default is <code>xterm -e (cmd /K)</code>.</p>

GST-specific environment variables

This section lists and describes all Genero Studio specific environment variables.

See the topic, Genero environment variables, in the *Genero Business Development Language User Guide* for a listing and definition of Genero specific environment variables such as *FGLDIR*, *FGLGUI*, and *FGLPROFILE*.

GREDEBUG

Defines the debug level for the Genero Report Engine.

Valid values include:

- 0: nothing
- 1: fatal only
- 2: fatal and error
- 3: fatal, error, and warning
- 4: fatal, error, warning, and info
- 5: fatal, error, warning, info, and debug
- 6: fatal, error, warning, info, debug, and trace
- 6+: all debug information

GRE_DEFAULT_IMAGE_URL

Defines the URL of the fallback image for an Image Box in a report design document.

The *fallback image* is the image to display if the requested image for an Image Box is not found.

The URL can be an absolute or relative URL. If it is a relative URL, the URL is resolved relative to the location of the form design (4rp) document.

This environment variable is specific to Genero Report Writer.

GREDIR

Defines the installation directory of the Genero Report Engine.

GSTDIR

Defines the installation directory of Genero Studio.

GSTLIBRARYDIR

Defines the location of the libraries used with the database generation script. By default this environment variable is set in the **Studio Libraries** environment set.

GSTSCHEMANAMES

Defines the schema files (4db, 4dbx, sch) to be loaded by default into the Meta-schema Manager when Genero Studio is launched. Available meta-schemas are displayed in the DB Schemas tab, and available to Genero Studio components such as Form Designer.

Use the GSTSCHEMANAMES environment variable to specify global schemas.

Note: Although you can specify global schemas, it is recommended that you add schemas to projects instead. Schemas added to projects are loaded when the project is opened, not at Genero Studio launch. Project can also be available to all developers without any additional configuration needed.

Select the default environment set (**Global Database Schemas**) or create a new one that includes the GSTSCHEMANAMES environment variable. Set the GSTSCHEMANAMES environment variable to specify the file names of the schemas to make available. Use the Value List environment variable type to list multiple schemas, separated by semi-colons. Do not include the file extension.

Use FGLDBPATH to define the directories in which to find the schema files listed in the GSTSCHEMANAMES variable.

GSTSETUPDIR

Defines the BAM application generator template directory. Changing this variable launches synchronization from the server and reloads the templates.

Select the default environment set or create a new one that includes the *GSTSETUPDIR* specifying the location of the template directory to be used.

GSTUSERSAMPLESDIR

Defines the Genero Studio samples directory for the user.

The *GSTUSERSAMPLESDIR* environment variable indicates the installation directory of the Genero Studio demo samples.

It is typically used when setting other environment variables.

By default, the *GSTUSERSAMPLESDIR* is not used in Genero Enterprise v3.0.

GSTWCDIR

Defines the directory in which the WebComponent XML files (*.wcsettings*) and optional image files are stored. By default this environment variable is set in the **Web Components** environment set.

Set *GSTWCDIR* to your web component directory where the *.wcsettings* and the optional image files reside. The Web Components files themselves may reside in a separate directory for deployment.

Each of the *.wcsettings* XML files describes a single [WebComponent](#) object.

Once you have set this directory, you may add a [WebComponent](#) widget to your form design document. The components described in the *.wcsettings* files will be available in the combobox list of the [componentType](#) property in the Properties view, allowing you to specify the particular WebComponent you wish to add to the form.

Add or edit environment variables

The **Environment Variable** dialog is used to add and edit environment variables.

When the **Environment Variable** dialog appears, enter:

Type	The type of environment variable. Options are Value, Value List, Directory, Directory List, File, or File List.
Name	The name of the environment variable.
Value	<p>The value of the environment variable. When entering the value, if the type is Value List, Directory, Directory List, File, or File List, select the ellipses (...) to browse for the correct value.</p> <p>If the value contains a variable name, that name must be prefaced with \$ and enclosed in parenthesis; for example \$(FGLLDPATH).</p> <p>The list separator is always a semicolon (;) on all systems (Windows™ and UNIX™). The directory separator in a path is always a slash (/) on all systems.</p> <p>Tip: Use the semicolon to separate directories in a list, and the slash (/) as the separator in a path, for portability of projects across operating systems.</p>

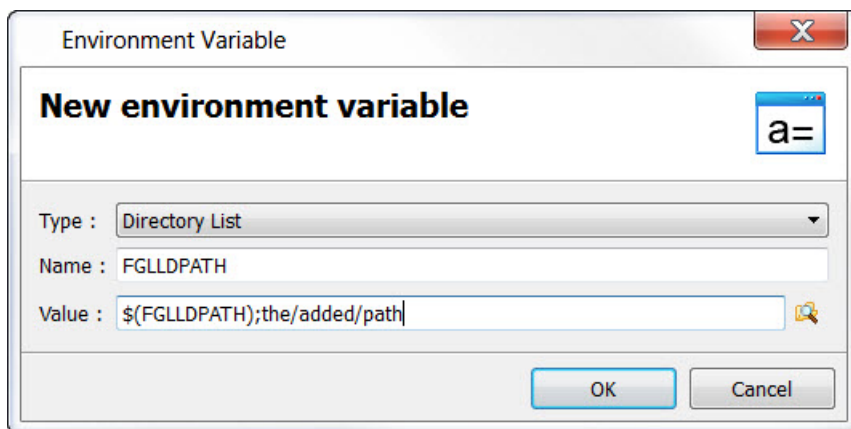


Figure 93: Setting FGLLDPATH

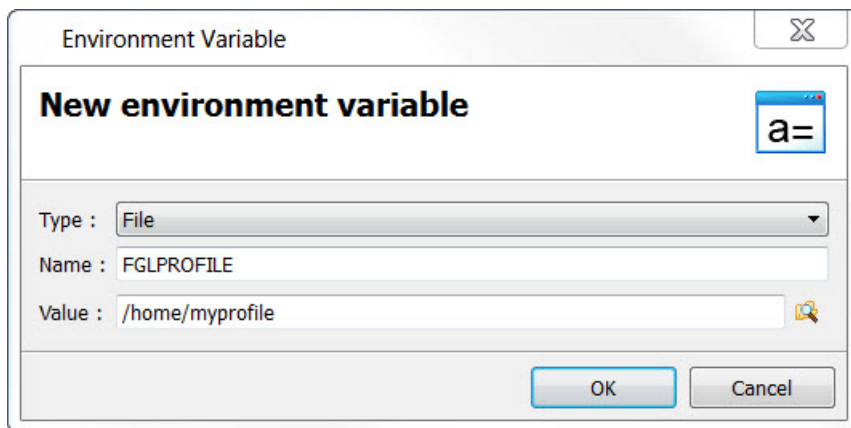


Figure 94: Setting FGLPROFILE

Reusing existing environment variables

A variable defined for a parent or ancestor node can also be reused in definitions for a child node:

For example:

- The parent node defines: `MY_VALUE=hello`
- The child node can reuse the parent node variable: `MY_COMPLETE_VALUE=${MY_VALUE} world`
- The final value of `MY_COMPLETE_VALUE` is "hello world".

For example:

- The parent node defines: `MY_VALUE=foo`
- The child node can reuse the parent node value and redefine the variable: `MY_VALUE=${MY_VALUE} bar`
- The final value of `MY_VALUE` will be "foo bar".

As a result, the System environment variables or Genero Studio Configuration variables can be reused in User Variable definitions within Project Manager.

Display clients

You can configure a variety of display clients in Genero Studio.

Desktop: GDC configurations

GDC configurations contain information about the available Genero Desktop Clients (GDC).

You can create multiple GDC Configurations, each with a different name.

Note: For information on installing the Genero Desktop Client outside of Genero Studio, see the *Genero Installation Guide*.

Configure for the Genero Desktop Client - Select Display Client

In the Display section of the Genero Configuration Management dialog, select the **Use Display Client** radio button.

The screenshot shows a dialog box titled "Display". It contains three radio buttons: "None", "Use Display Client", and "Use GWC". The "Use Display Client" radio button is selected. Below the radio buttons are three dropdown menus: "Use Display Client" (set to "GDC 3.00.00"), "Use GWC" (set to "<Choose a GWC>"), and "Application Type" (set to "Web Application"). There are also two more dropdown menus: "Theme" (empty) and "Browser" (set to "<Default Browser>"). There are small edit icons next to the "Use Display Client" and "Browser" dropdowns.

Figure 95: Use Display Client (Genero Desktop Client) for the display

Display Client configuration combobox

Select the display client configuration to use.

To add a new GDC configuration, click the Edit icon next to the Display Client configuration combobox. The **Display Client management** dialog opens.

Configure for the Genero Desktop Client - Display Client management

To set a GDC configuration, select GDC from the **Client type** combobox.

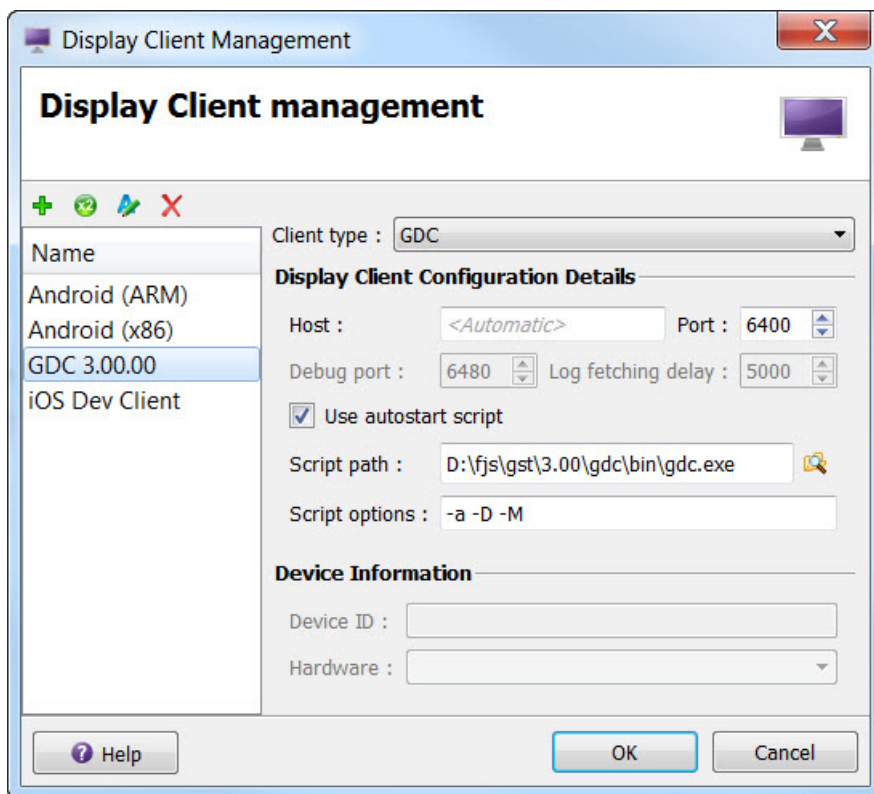


Figure 96: Display Client Management dialog

Select a configuration from the list to modify its settings. Use the integrated Toolbar to add, duplicate, remove, or edit a configuration. Once a name is added, enter its settings.

The following fields apply to a GDC configuration:

Host

Hostname or IP address where the GDC client executes. In local host mode, when the host field is empty, the client IP address will be automatically detected in order that GDC will always be accessible from fgllrun. You can change this manually.

Port

Port from which GDC client is to be launched.

Use autostart script

Check this option to have Genero Studio automatically start the GDC.

Uncheck this option if you want to manually start the GDC.

Script path

Path to where GDC client is installed.

Script options

Command line options for the execution of Genero Desktop Client (GDC). For a full list of valid command line options, see the *Genero Desktop Client User Guide*.

Web: GAS/GWC configurations

To display web applications or to run web services, you must configure both for the application and for the Genero Application Server.

You can create multiple GAS Configurations, each with a different name.

Note: If you did not install the entire Genero Studio package initially, you must install and set up the Genero Application Server (GAS), which includes the GWC rendering engine. For information on installing the Genero Desktop Client outside of Genero Studio, see the *Genero Installation Guide*.

When you select a GAS configuration, Genero Studio will dynamically generate an external application to enable you to run, debug, or preview an application or file.

For development, the [GAS standalone dispatcher: httpdispatch](#) on page 174 can be used instead of a web server.

You can specify the browser to use with GWC in [General Preferences](#) on page 106 or in the [Configuration Management](#) dialog.

Configure for the Genero Web Client

In the Display section, select the **Use GWC** radio button.

The screenshot shows a configuration window titled "Display". It contains several options:

- None
- Use Display Client : <Choose a Display Client>
- Use GWC : GAS 3.00.00
- Application Type : Universal Agent
- Theme : (empty dropdown)
- Browser : <Default Browser>

Figure 97: Use GWC for the display

GAS configuration combobox

Select which GAS configuration to use. To add a new GAS configuration, click the Edit icon. The **Genero Application Server Management** window opens.

Application type

Select `Universal Agent` to use GAS 3.00 GWC-JS.

Select `Web Application` to use GAS 2.50 GWC-HTML5 (deprecated).

Theme

Select a theme to use.

Browser

Specify a specific browser to use. When left blank, the default browser is used.

Configure the Genero Application Server

In the Genero Application Server Management window, existing GAS configurations are listed in the left-hand panel. Select a configuration from the list to modify its settings. Use the integrated Toolbar to add, duplicate, remove, or edit a configuration. Once a name is added, enter its settings.

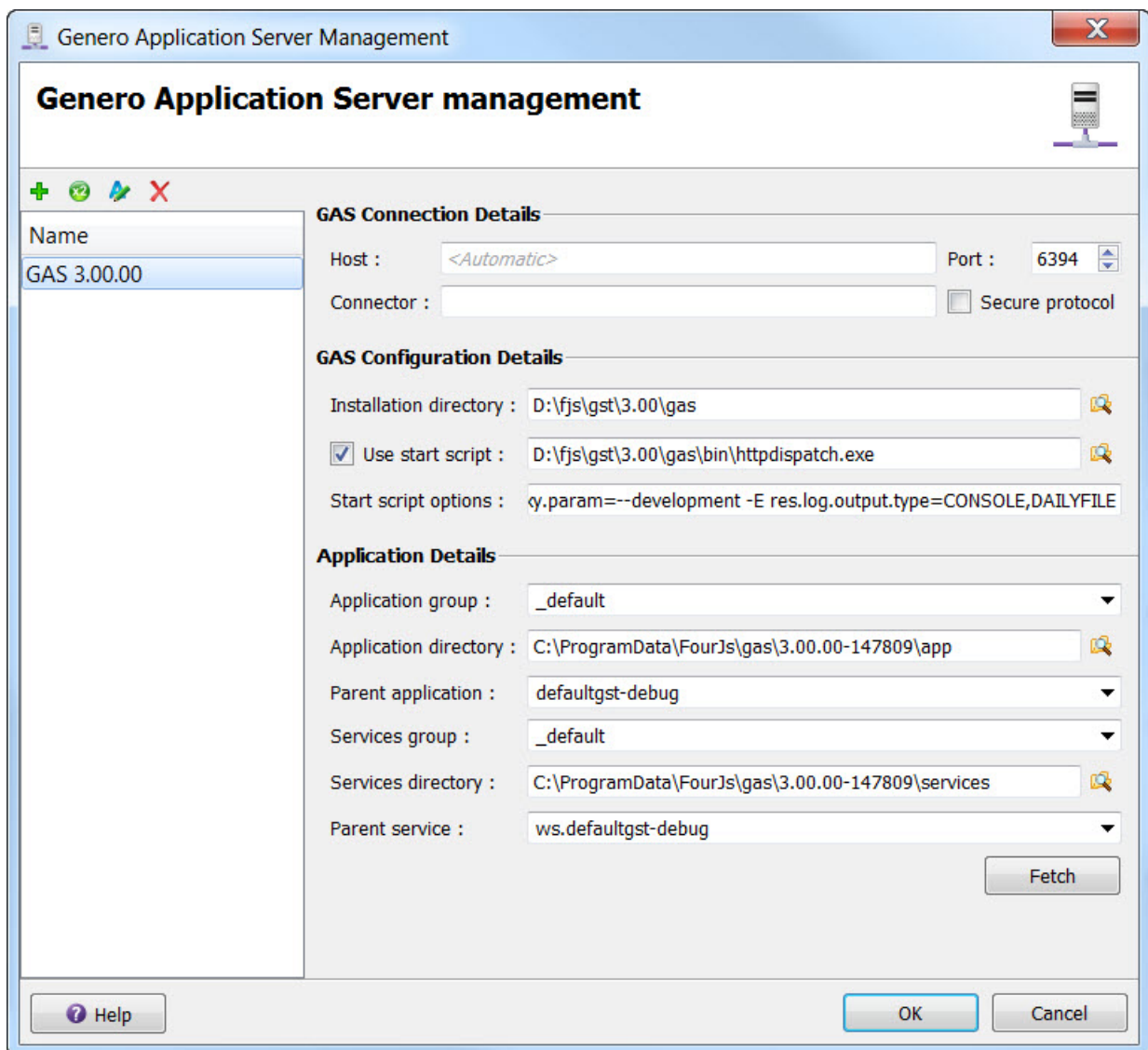


Figure 98: GAS Configuration

GAS Connection Details section

Host

Specify the IP address or the network name of the server you are connecting to when running an application with the GAS. If you are in a “direct connection” using GAS Standalone, specify the IP address or the network name of the Application Server (the one where the compiler resides). If you are connecting through a web server, specify the IP address or the network name of the web server.

Port

Specify either the port of the GAS (if you are in a “direct connection”, GAS Standalone) or of the web server.

Connector

Leave blank if you are using a “direct connection” using GAS Standalone for development purposes. Enter the alias configured for your dispatcher in your GAS installation (often, `gas`). See the section, “Configuring the Genero Application Server” in the

Secure protocol

Genero Application Server User Guide for more information on configuring your dispatcher.

The URL used when launching an application is impacted by this selection.

- When selected, the protocol is "https".
- When not selected, the protocol is "http".

GAS Configuration Details section**Installation directory**

Directory where the GAS is installed locally.

Use start script

The check box specifies whether to use the startup script to launch the GAS. When checked, the field specifies which start script to be launched when this configuration is used. The [GAS standalone dispatcher: httpdispatch](#) on page 174 can be specified here to launch `httpdispatch`. `httpdispatch` can be used as the GAS script. In versions prior to 2.50, set the environment by specifying `-p <GASDIR>` in the **Start script options** field. Alternatively, call a script that calls `./envas` before calling `httpdispatch`, such as:

```
./envas
httpdispatch
```

For GAS versions prior to 2.30, set the path for `gasd` instead of for `httpdispatch`.

Note: The start script is only used when no GAS is found on the host at the port specified. The start script is not called even if the GAS to be launched from start script is different from the one already launched.

Start script options

Specify any options to be used with the start script. See [GAS standalone dispatcher: httpdispatch](#) on page 174 for `httpdispatch` options. For more information on script parameters, see the *Genero Application Server User Guide*.

Application Details section**Application group**

The name of the Application Group defined in the GAS `as.xcf` file. Enter the value `_default` for the group to use the default configuration file, unless you have an explicit group to specify.

Application directory

The directory of the Application Group, the same path as the one referred to in the `as.xcf` file. If using the default application, the corresponding directory (usually `GAS installation directory/app`) has to be entered in the **Group Directory** field.

Parent application	The identifier of the parent GWC application; if no identifier is given, the default GWC application will be used.
Services group	The name of the Web Services Group defined in the GAS <code>as.xcf</code> file. Enter the value <code>_default</code> for the group to use the default configuration file, unless you have an explicit group to specify.
Services directory	The directory of the Web Services Group, the same path as the one referred to in the <code>as.xcf</code> file. If using the default application, the corresponding directory (usually <code>GAS installation directory/app</code>) has to be entered in the Group Directory field.
Parent service	The identifier of the parent Web Services application; if no identifier is given, the default Web Service application will be used.

URL example

The URL used when launching application from Project Manager is impacted by the **Secure protocol** option and the **Application Type** selection:

```
<protocol>://<host>:<port>/<connector>/<application_type>/r/
<application_name>
```

- `protocol` is either "http" or "https", depending on the value of the **Secure protocol** option.
- `application_type` is either "ua" or "wa", depending on the value of the **Application type** option.
- `application_name` depends on the executed application in project manager.
- Other values depend on the GAS configuration parameters.

Example GAS configuration

- **Host:** `<Automatic>`
- **Port:** `6394`
- **Application group:** `mygroup`
- **Group directory:** `c:\work`
- **Parent application:** `defaultgwc`

In this configuration, GAS applications will be run using the GAS located on the current computer at port "6394". The group "mygroup" in the `as.xcf` file points to the directory "c:\work"; the corresponding fields in the GWC configuration dialog are filled with the same information.

Note: Users on Windows™ 64-bit machines who are using a network proxy: The browser cannot open 127.0.0.1 or localhost unless you modify your Advanced Network settings to avoid going through the proxy for these addresses.

See the *Genero Application Server User Guide* for information on setting up GWC and GAS and adding applications.

Create and apply a custom XCF for your Web application

Follow these steps to create and apply a custom configuration for your Web application within the Genero Studio framework.

When launching a Web application, information needed to launch that application is provided in an external application configuration (`XCF`) file. An `XCF` file is an external `XML` file that provides the configuration details for an application being launched by the Genero Application Server (GAS). When Genero Studio

launches a GWC application, it generates the XCF file and places it in the default directory for application configuration files, as specified in the GAS configuration file. If you have explicit changes that need to be made to the generated XCF file - such as specifying a specific snippet to use or parameters to pass to the application - you must create a custom XCF file.

Note: Refer to the GAS documentation for information on creating an XCF file and launching Web applications outside of Genero Studio.

1. Create a custom application configuration XCF file. Use Code Editor to create the file as it provides syntax highlighting and auto-completion while editing. Validation is done using the corresponding XSD, and any errors are displayed.

Include only those elements that you need to add for customization purposes.

Example custom XCF file (myHelloWorld.xcf):

```
<?xml version="1.0" encoding="UTF-8" ?>
<APPLICATION Parent="defaultgwc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.4js.com/ns/gas/2.10/
  cfextwa.xsd">
  <EXECUTION>
    <PARAMETERS>
      <PARAMETER>--test</PARAMETER>
      <PARAMETER>-o</PARAMETER>
      <PARAMETER>outputFile.txt</PARAMETER>
    </PARAMETERS>
  </EXECUTION>
  <OUTPUT>
    <MAP Id="DUA_SL" Allowed="TRUE">
      <THEME>
        <TEMPLATE Id="_default">c:\mypath\main2.xhtml</TEMPLATE>
        <SNIPPET Id="ComboBox">c:\mypath\SL\combobox.xaml</SNIPPET>
      </THEME>
    </MAP>
  </OUTPUT>
</APPLICATION>
```

2. Add the custom XCF file to your application's project structure.

Your file can have any valid file name, and must end with an xcf extension. Add the file to an application node or dependent library.



Figure 99: Project View with Custom XCF file displayed

3. Launch the application in Genero Studio.

Genero Studio searches for a custom XCF file first in the application node. If there is no custom XCF file in the application node, Genero Studio will use the first custom XCF file found in the libraries that are dependencies of the application. If multiple custom XCF files are located, the first found file is used. If no custom configuration file is found, the default configuration file created by Genero Studio is used.

The information in the found custom XCF file is applied to the generated XCF file. Two XCF files are automatically generated when the application is run with the GWC client.

application_user.xcf

This file is intended to be re-used afterward in a normal GAS/GWC configuration.

application_user_studio.xcf

This file is temporary and only used by Genero Studio for running the web application for a single

session; it is not intended to be (modified and) re-used afterward.

Mobile clients: GMI and GMA configurations

Android and iOS configurations contain information about the available Genero Mobile clients (GMA and GMI).

See the *Genero Mobile Developer Guide* for steps to display your app to an Android or iOS device or emulator.

In the Display section, select the **Use Display Client** radio button.

To set a mobile configuration, select the desired mobile option from the **Client type** combobox.

To open the Display Client Management dialog, click on the Edit icon.

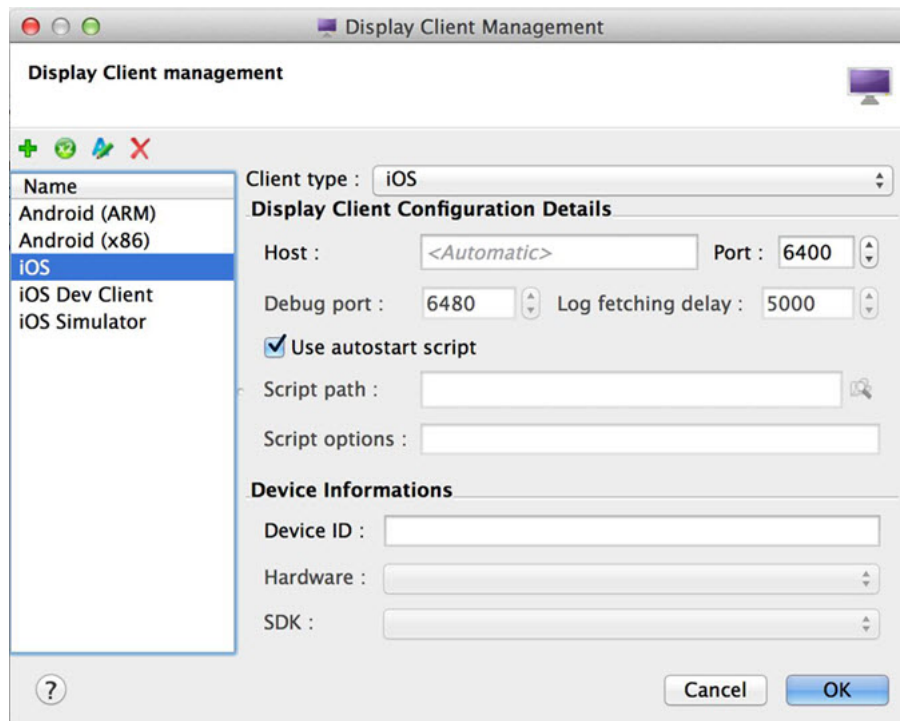


Figure 100: Display Client Management dialog

Select a display client from the list to modify its settings. Use the integrated Toolbar to add, duplicate, remove, or edit a configuration. Once a name is added, enter its settings.

Display Client Configuration Details

Client type

Select the client type for the configuration.

Host

Hostname or IP address where the mobile client executes.

Port

Port from which the client is to be launched.

Debug port

Port for displaying debug information.

Log fetching delay

The delay in milliseconds before a log is fetched.

Use autostart script

If checked, autostart script will be executed if client is not running.

Script path	Path of script to execute when Use autostart script is checked. For GMI and GMA clients an implicit script is used.
Script options	The options passed as parameters to the autostart script. Not used with GMI and GMA.

Device Informations

Settings for identifying display devices so that multiple display devices can be connected at the same time. See *Configure multiple display devices* in the *Genero Mobile Developer Guide*.

Device ID	The identifier of device (iOS or Android). Use the List Devices option in the Tools >> Android Tools or Tools >> iOS Tools menu to get list of device IDs. Tip: If you are using iOS and Xcode, you can use a substring from the string displayed by the List Devices option. For example, you can use the label of the device as displayed in the General settings.
Hardware	Android only, specifies the architecture of the target CPU, x86 or ARM .
SDK	Currently only used for iOS Simulator configuration. The only possible value is iOS Simulator 7.0 .

Setting up a remote environment

These topics assist you in setting up a remote Genero Studio environment.

- [Remote environment software requirements](#) on page 154
- [Add a remote host](#) on page 155
- [Define mount points to shared drives](#) on page 161
- [Define remote Genero installations, GAS configurations, and environment sets](#) on page 162

Remote environment software requirements

In a remote environment, installation is done on the remote server and accessed by developers from their client machines.

On the server(s)

- Installed Genero Studio server side tools. See your installation documentation.
- Installed relational database (Informix®, Oracle, etc.) and its corresponding client (CSDK, Oracle Client, etc.); **or** Database Client on this server and the corresponding Database on a different remote server.
- SSH server (if using the recommended [SSH connections](#)).

On the client

- Genero Studio
- Installed Genero Desktop Client

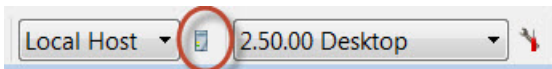
Each database software has its own client character set configuration. You must properly configure the database client locale in order to send/receive data to the database server, according to the locale used

by your application. See Localization: Database Client Settings in the *Genero Business Development Language User Guide*.

Add a remote host

To connect a Genero Studio client to a remote Genero Studio Server, add the remote server to the **Hosts** defined in Genero Studio Configurations and define a remote configuration for that server.

1. Start Genero Studio on the client.
2. Select **Tools >> Server connections** or select the **Server Connections** button in the lower right corner.



3. Select the button in the integrated Toolbar to **Add a Genero Studio server**.

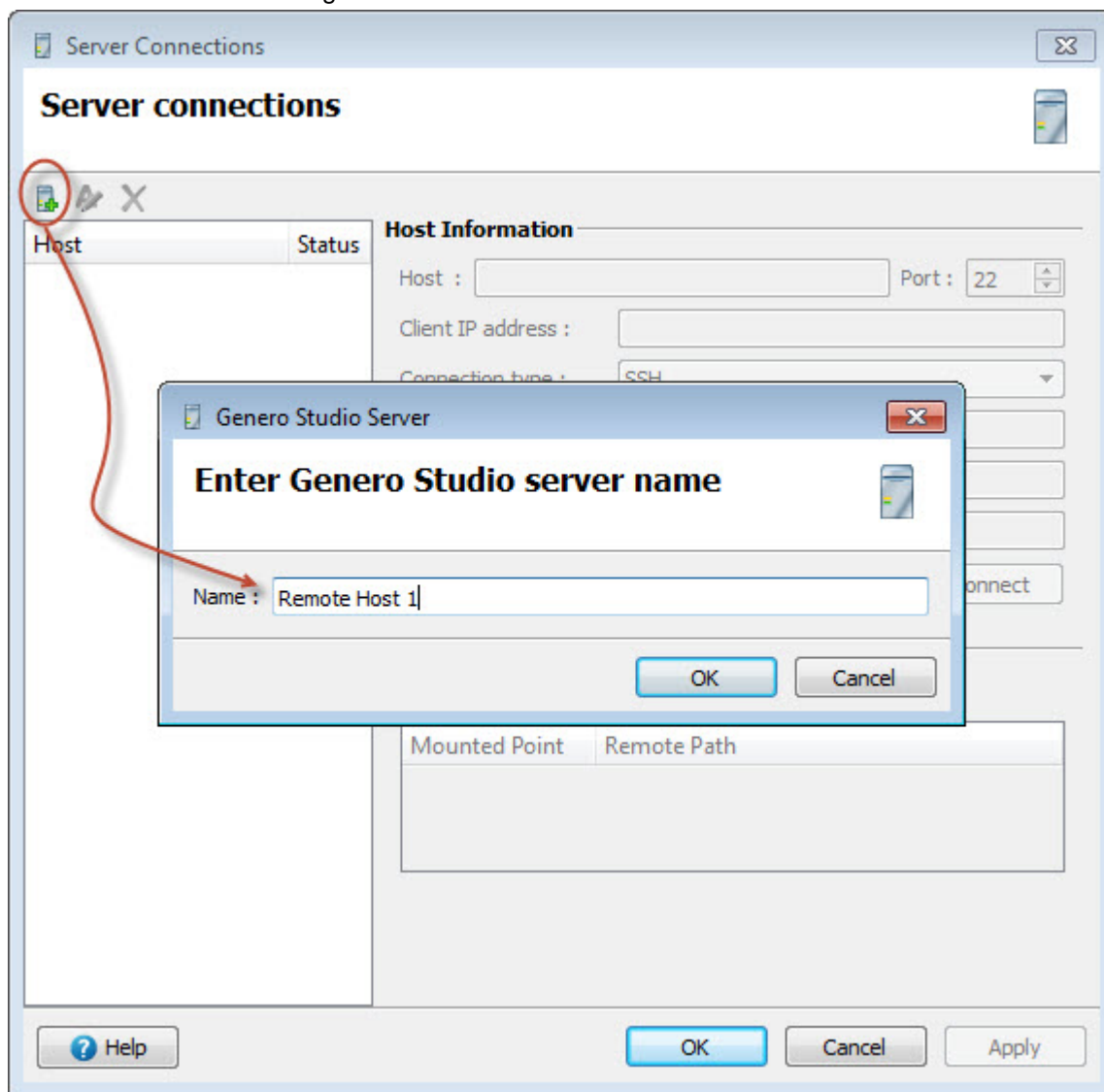


Figure 101: Add a Genero Studio server

4. Enter the **Host Information**.

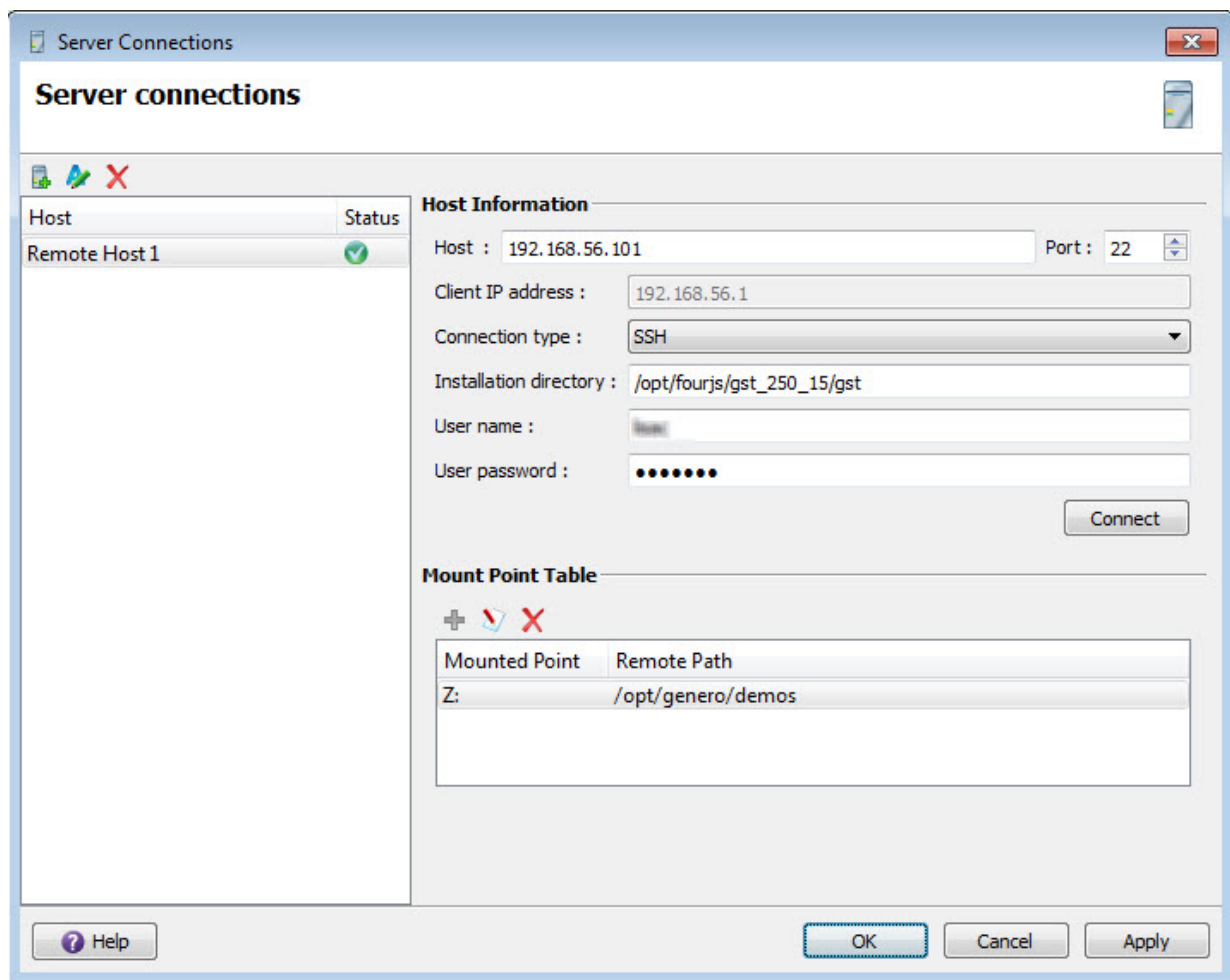


Figure 102: Host Information

Host	Hostname or IP address where Genero Studio Server is running.
Port	Enter the port number you have configured SSH to listen on. The default SSH port number is 22. If you are using TCP (deprecated), enter the port number used when Genero Studio Server was started. The default port number is 5321.
Client IP address	IP address of the client as viewed by the server. Displays the value of the IP address to be used by the GDC settings when <i><Automatic></i> is set.
Connection type	SSH (default) or TCP (deprecated).
Installation directory	The Genero Studio installation directory on the remote machine; enter the complete path, which will always end with /gst
User name	Valid user name. See SSH advanced security on page 157.
User password	Valid password.

5. Select **Connect** to validate the host configuration.

6. Once a valid connection has been made, existing Genero installations, GAS configurations and Environment sets can be imported from the server. Select the **Import** button. See [Import Configuration dialog](#) on page 173.
7. [Define mount points to shared drives](#) on page 161.
8. Select OK. Your new remote host will now be listed in the host list in the lower right corner of Genero Studio.

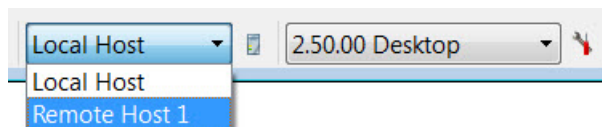


Figure 103: Host list

SSH

Each user with an SSH connection works with their own server process, as one instance of Genero Studio server is created for each SSH connection. If you choose this connection type, an SSH daemon must be running on the host. There is no need to launch Genero Studio Server on the host; **the SSH connection will automatically launch it..** On Windows™, a third party SSH server must be installed.

SSH advanced security

- [Configuring Public Key Authentication on Windows™](#)
- [Configuring Public Key Authentication on Linux™](#)

Configure public key authentication on Windows™

Genero Studio can support public key authentication on Windows™ if you use **Pageant**. You have to install **PuTTY** on your client machine to use it. **Pageant** is a utility that stores unencrypted private keys in memory and provides digital signatures when needed.

You configure it by giving the path of the encrypted private key file (See PuTTY documentation for more information). You enter the passphrase just once when Pageant is launched; then you will not have to bother with it again, until you launch a new Windows™ session.

Genero Studio uses **plink** to initiate an ssh connection. If Pageant is launched, plink automatically uses Pageant to get a signature. When using public key authentication in Genero Studio, don't fill in the password field in the SSH connection dialog, as it not used.

Pageant

First, you have to generate a pair of private\public keys. If you already have a SSH\OpenSSH key pair, see the [How to use openSSH key with Pageant](#) section.

- Launch PuTTYgen, and verify that the **SSH-2 RSA radio button** is selected.
- Select the number of bits you want as the size for the key and click the **Generate** button. During generation, move cursor into the blank area to randomize the keys.
- Enter a comment to identify the key pair more easily.
- Enter a passphrase. A passphrase is used to decrypt the private key (Private key is encrypted on the disc for more security). Don't forget it, you will have to enter it each time you begin a new Windows™ session.
- Once you have entered the passphrase, you can save the private and public keys on the disc.

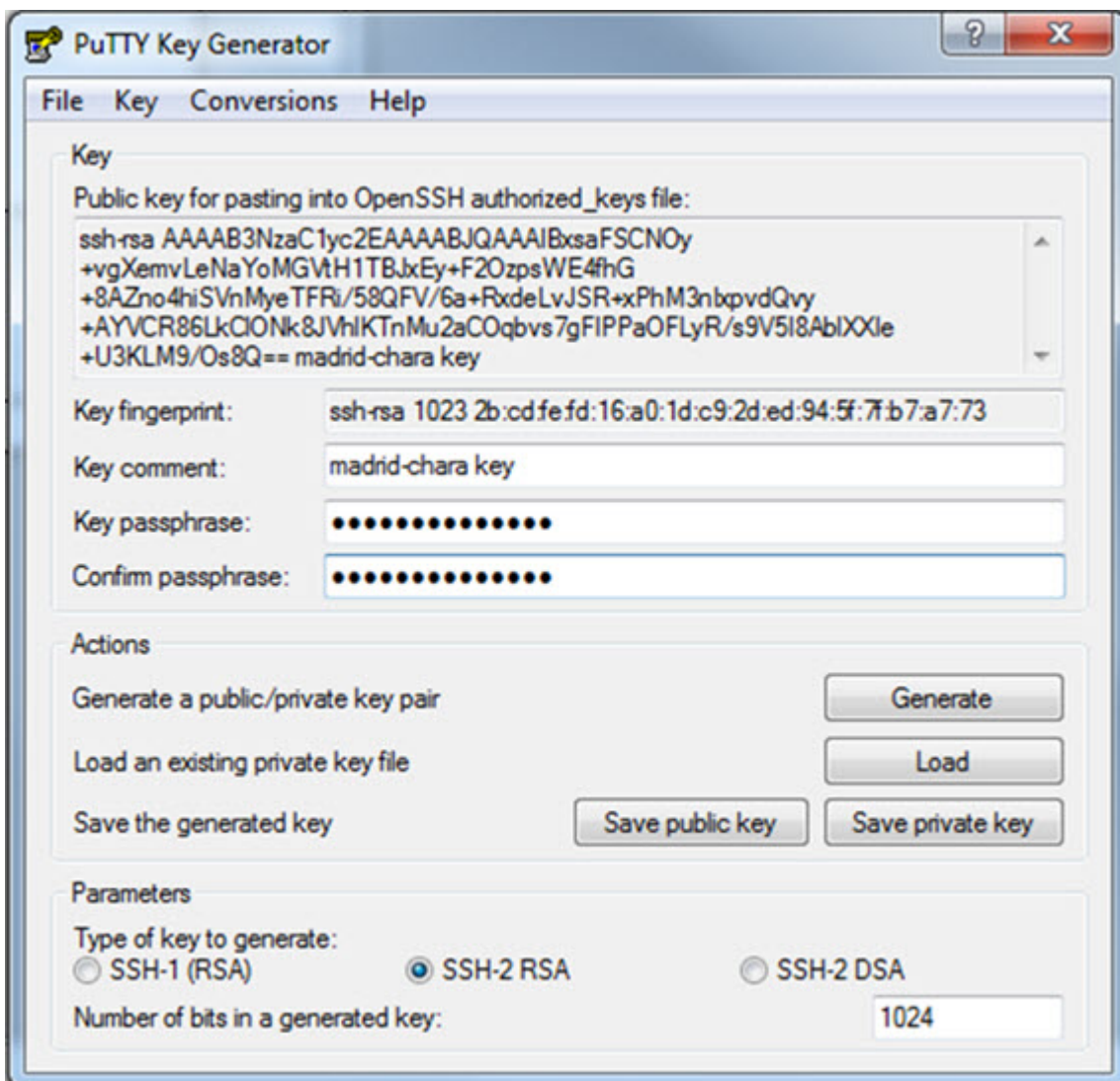


Figure 104: PuTTY Key Generator

- Next, go to your home directory (`/home/username`) on your host machine, and edit the `.ssh/authorized_keys` file with your preferred text editor. If the file (or `.ssh` directory) doesn't exist, create it.
- Copy your generated public key in the `authorized_keys` file. The key must be in exactly the same format as displayed in PuTTYgen (`ssh-rsa ****...****== keyname`) and must occupy only one line. If several lines were already present, enter a line return before writing the key.
- Now launch Pageant. It will appear in the systray as a desktop computer wearing a hat. Right click on it and select **Add key**.

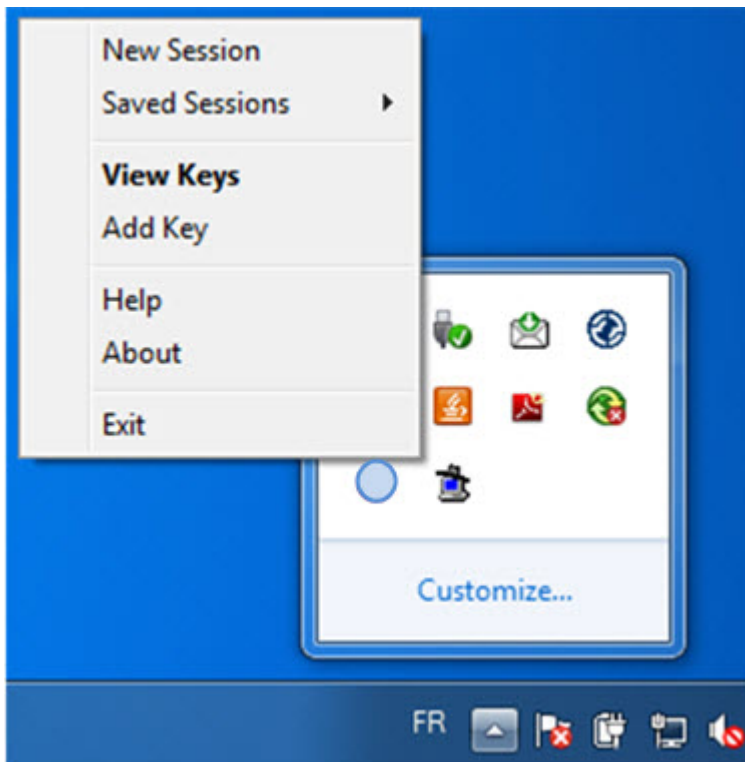


Figure 105: Pageant pop-up menu.

- An open file dialog lets you select a ppk (Private key) file. After selecting it, you will be asked to enter the corresponding passphrase.
- You can verify that the key has been successfully added by selecting "View keys" in the Pageant icon's context menu. The key signature is displayed instead of the private key, but you can recognize it by its name.

You should now be able to initiate an ssh connection with Genero Studio without entering a password.

How to use openSSH keys with Pageant

The ssh-keygen generated private key files cannot be directly used by Pageant; you have to convert them into .ppk files. To do so, open PuTTYgen and select **Conversions>>Import key menu item**; select your OpenSSH format private key.

Configure public key authentication on Linux™

Genero Studio uses **plink** ssh client to initiate an ssh connection; plink is traditionally used with the Pageant authentication agent. Since Pageant doesn't exist on Linux™, we will use **ssh-agent**, which is released with OpenSSH clients.

ssh-agent stores the private key unencrypted in memory, and provides the private key signature when asked. You configure it by giving it the path of the encrypted private key file (using the ssh-add command). When ssh-agent is launched, plink will automatically ask ssh-agent to get a signature to send to ssh server.

ssh-agent

First, you have to generate a private\public key pair. If you already have a PuTTY generated key pair, see the [How to use PuTTY keys with ssh-agent](#) section; otherwise, we will generate it with ssh-keygen.

- Type **ssh-keygen -t rsa** in a terminal to generate a key pair using rsa encoding.
- You will be prompted for the file path where you want to save your private key. If you press enter without giving a path, the key will be saved as **~/.ssh/id_rsa**.

- Next, enter the pass phrase (needed to decrypt the private key). Don't forget it as you will have to enter it again at each logon session. The public key file will be saved with the same path as the private key file, but with the .pub extension.
- Go to your home directory (`/home/username`) on your host machine, and edit the `.ssh/authorized_key` s file with your preferred text editor. If the file (or .ssh directory) doesn't exist, create it.
- Copy your generated public key in authorized_keys file. The key must be exactly in the same format as displayed in PuTTYgen (ssh-rsa ****...****== keyname) and must occupy just one line. If several lines were already present, enter a line return before writing the key.

Next, configure **ssh-agent** to automatically provide signatures from the private key.

- **ssh-agent** must be started at each logon session. Execute it if it isn't already running.
- Next, execute **ssh-add** with the private key file path as the parameter (For example **ssh-add ~/.ssh/id_rsa**). This will tell **ssh-agent** to test your private key each time you start an ssh session.
- Verify that the key has been successfully added by typing **ssh-add -l**. It should display you your private key fingerprint.

You can now use public key authentication with Genero Studio. For that, create an SSH connection without entering a password:

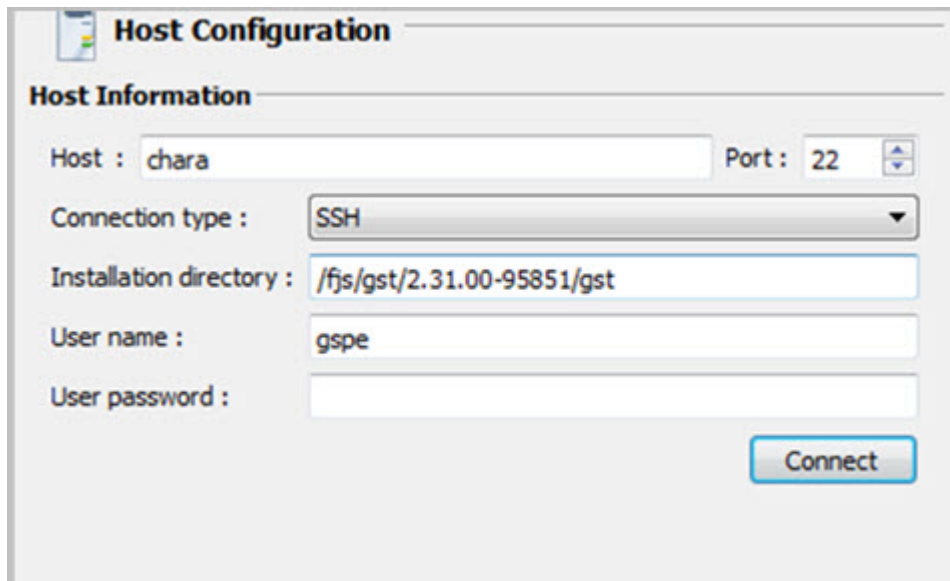


Figure 106: Host Configuration

How to use PuTTY keys with the ssh-agent

If PuTTY is installed on Windows™, start PuTTYgen, load your private key (ppk file), and select the **conversions >> Export OpenSSH key** menu item. It will ask you where you want to save the file.

If PuTTY is installed on Linux™, execute **puttygen input_key.ppk -O private -o output_key**.

SSH troubleshooting

This topic covers common SSH issues and how to resolve them.

Table 22: Errors when trying to connect to a server using SSH

Type	Issue	Resolution
Trying to connect to a server using SSH	<code><installation directory></code> : No such file or directory.	The path given in installation directory field is incorrect.

Type	Issue	Resolution
Trying to connect to a server using SSH	Access denied.	Either the user name or password is incorrect.
Unable to open a connection	Host does not exist.	The host either does not exist or is not reachable.
Unable to open a connection	Network error: Connection refused.	Verify that the port number corresponds to the SSH server's one.
Unable to open a connection	Network error: Connection reset by peer.	This happens when the networked machines lose the connection between them. Try to reconnect.

TCP (deprecated): How to launch Genero Studio Server

If you use a TCP connection, Genero Studio Server must be launched before use. (If you use an SSH connection, Genero Studio Server is launched automatically.)

This program has no user interface; it must be run from a console or terminal. On the server machine, from within *GSTDIR/bin*, execute the command for your operating system. The default port value is **5321**.

- On Windows™:

```
gssserver.exe
```

- On GNU/Linux:

```
gssserver
```

If you need it to listen to another port, you have to use the command-line argument **--port** each time you start Genero Studio Server.

- On Windows™:

```
gssserver.exe --port 1234
```

- On GNU/Linux:

```
gssserver --port 1234
```

For further information, start Genero Studio Server with the **--help** option, and see the **installation notes in the release subdirectory** of the Genero Studio software package.

Define mount points to shared drives

Shared drives allow you to compile, run, and debug files that are on the remote server.

This task assumes you have [added a remote host](#).

Export the file system on your server using Samba / NFS / Windows™ Share. You must share the drive using a standard tool, such as NFS (Network File System) for Linux™ networks. Samba, or a similar tool, can be used to allow Microsoft™ networks to share files stored on a Linux™ server.

All program files accessed remotely must reside on a shared drive on a server accessible to the client's local network. Genero Studio uses mapped paths to allow simultaneous access to two machines.

1. Open the Genero Studio Server dialog and select a remote host from the list.

2. Select the **Add a mount point ...** button from the **Mount Point Table** integrated Toolbar. A dialog lists the mount points available for this server.
3. Select a mount point and enter the remote path.

Windows™ Genero Studio clients:

Mounted point	The mounted drive, the network drive that has been mapped to a letter drive on your local machine.
Remote path	The related remote path. For example, if your local computer has a mounted drive G: that is mapped to <code>/usr1/public</code> on the server zebra, the remote path for drive G: is <code>/usr1/public</code> .

Linux™ Genero Studio clients:

Mounted point	The location of the file system where the device is attached.
Remote path	The related remote path. For example, if your local computer has a mount point <code>/mnt/work</code> that is mapped to <code>/usr2/public</code> on the server zebra, the remote path for <code>/mnt/work</code> is <code>/usr2/public</code> .

4. Select **Apply** to save your changes.
The character set will be specified in the environment of the **remote** FGL installation (set the **LANG** variable in the **remote** configuration variables). On the client machines, set [Language support \(text encoding\)](#) on page 163, specifying the same character set.
5. [Define remote Genero installations, GAS configurations, and environment sets](#) on page 162.

Define remote Genero installations, GAS configurations, and environment sets

Genero installations, GAS configurations, and environment sets can be defined for the remote host.

This task assumes you have [added a remote host](#).

1. Select the remote host from the list in the lower right corner of Genero Studio.
2. Select the configuration button to display the **Genero Configuration Management** dialog for the selected server.
3. Add new configurations as needed to add [Compiler / Runtime configuration \(Genero Installations\)](#) on page 172, [Environment sets](#) on page 140, and [Web: GAS/GWC configurations](#) on page 147.
4. Save the changes.

Share projects / source code management

Genero Source Code Management (SCM) enables collaborative sharing and maintaining of the files in Genero projects.

You can share your Genero projects using a Version Control System (VCS) such as Apache's Subversion:

- A network drive/directory must be set up on the remote server for the storage of the project and source files.
- All paths in the `4pw` files must be relative paths; only the path of the project(s) can be absolute.
- The network drive/directory must be mapped on each client machine.
- You will check out a copy of the project file and source files from the repository to your mapped network drive. After making changes, commit your files back to the repository.

A Subversion client must be installed on your local machine. Genero Studio for Windows™ includes Apache's Subversion client. Genero Studio for GNU/Linux relies on Subversion 1.6.2 or later, which must have been installed on the system. For more information about Apache's Subversion, see: <http://subversion.apache.org/>

To specify the location of your SVN client, select **Tools >> Preferences >> Source Code Management >> Subversion** from the main menu.

Access a database

To execute an application that accesses a relational database, Genero Studio uses the Genero runtime system and the specific database client software, which must be properly configured.

Entries in a FGLPROFILE configuration file can be used to connect to the database. See the specific page for the supported database vendor's software in the Open Database Interface section of the Genero BDL documentation.

Genero Studio [database meta-schema files](#) allow you to use the database tables and columns that are listed in the schema to define variables and form fields in your application code and form definitions.

Default environment sets are provided for each database vendor. Values for the environment variables, as well as additional variables, can be added in the [Genero Configuration Management dialog](#) on page 171 as part of the configuration process.

Language support (text encoding)

Text encoding specifies the character set to be used by Genero Studio. You can change the default encoding to work in your preferred language.

To support *internationalization*, characters typed at the keyboard are intercepted and changed automatically, based on the text encoding method selected.

A default character set is specified in **Tools >> Preferences, General Preferences**. This default text encoding method is the one found on your system. To use encodings for a different language, select a different text encoding choice from the list. In addition, text encoding plugins can be configured, which are loaded at startup to provide new encoding support.

Since the default encodings are much faster than plug-ins, users are strongly advised to use the provided encodings unless there are no other solutions.

The selected encoding is used during file operations of text documents (load, save, parsing, highlighting, and so on), so the encoding must be correctly set before opening documents. Once loaded, Genero Studio uses *Unicode* for text management.

- If you change the text encoding in Genero Studio, the corresponding `LANG` environment variable must be set.
- If you are using Genero Studio in a remote configuration, your encoding method must match the encoding on the Genero Studio server.
- If you are using CVS or other version control systems, the encoding method must be the same for all text stored in the system.
- The database client *locale* (character set) and application *locale* (character set) settings must match.
- Genero BDL and Genero clients have settings that must also be changed when you want to use encodings and character sets for a language that is different from that specified in the default text encoding for your system. See the topic on *Localization* in the *Genero Business Development Language User Guide* for additional information about the settings for Genero Business Development Language. See the topic on *Localization* in the *Genero Desktop Client User Guide* for additional information about settings for Genero clients.
- [Configure Genero Studio to use a different character set](#) on page 164

- [Add a text encoding plugin](#) on page 165
- [Character mapping table \(encodingMap.xml\)](#) on page 167
- [Configure keyboard and language on a Windows client](#) on page 168
- [Configure LANG on a Genero Studio Server](#) on page 168
- [Test text encoding configuration](#) on page 170

Configure Genero Studio to use a different character set

This example explains how to set up Genero Studio client and servers to use the Polish language with ISO 8859-2 encoding.

In this example, the Genero Studio client runs on a Windows™ 7 Ultimate 64-bit platform using English with the CP1252 character set (Western European languages). The Genero Studio server runs on enterprise Linux™ using English with the UTF-8 character set.

1. Configure the preferred encoding for the Genero Studio client. Select **Tools >> Preferences >> General** and locate the preferred encoding (ISO-8859-2 in this example) in the list of **Text Encodings**. If the preferred encoding is not available in the list, [add a text encoding plugin](#) to incorporate the preferred character set.

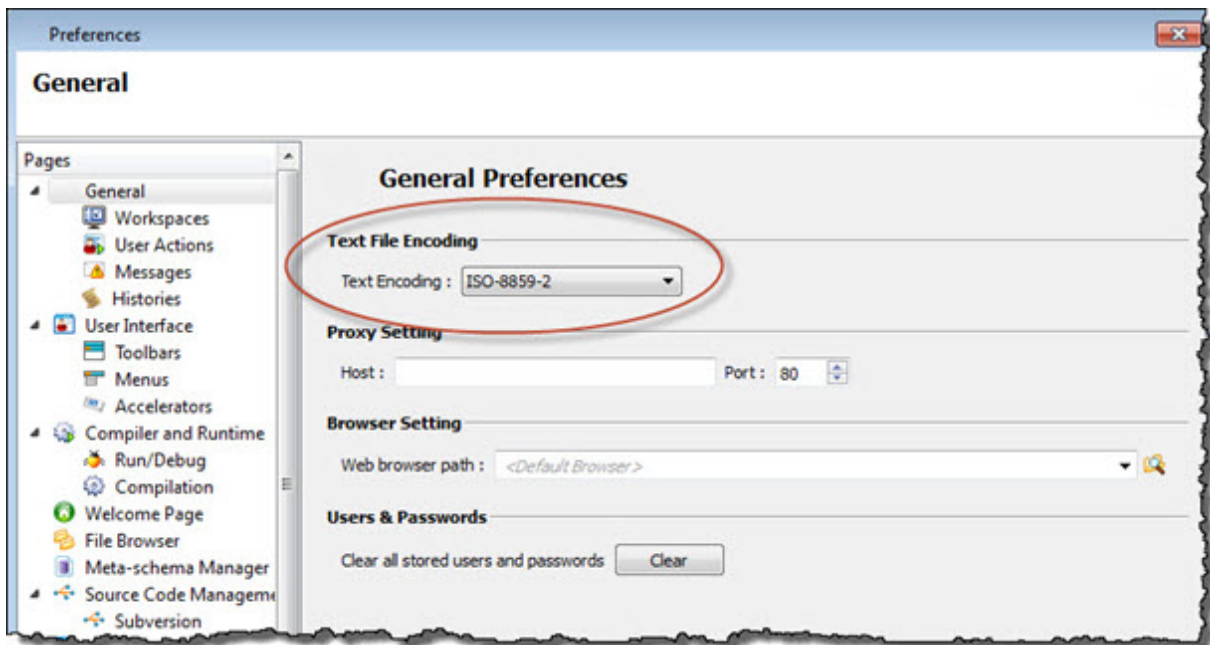


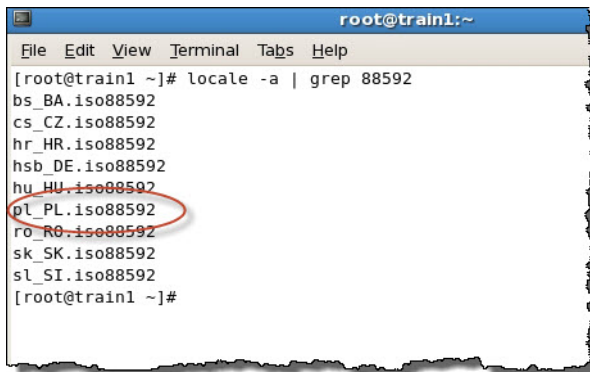
Figure 107: Text File Encoding in General Preferences

2. Check to see if the preferred encoding is supported on the Linux™ server (pl_PL.iso88592 for the example):

```
$ locale -a | grep 88592
```

If the preferred locale isn't present, install it per your system documentation. For example:

```
$ localedef pl_PL.iso88592 -i pl_PL -f ISO-8859-2
```



```

root@train1:~
File Edit View Terminal Tabs Help
[root@train1 ~]# locale -a | grep 88592
bs_BA.iso88592
cs_CZ.iso88592
hr_HR.iso88592
hsb_DE.iso88592
hu_HU.iso88592
pl_PL.iso88592
ro_RO.iso88592
sk_SK.iso88592
sl_SI.iso88592
[root@train1 ~]#

```

Figure 108: Checking available encodings with the locale command

3. Configure the desired keyboard and default input language on the Windows™ client. See [Configure keyboard and language on a Windows client](#) on page 168.
4. [Configure the selected language](#) for the Genero Studio Server environment.
5. [Test](#) the new configuration.

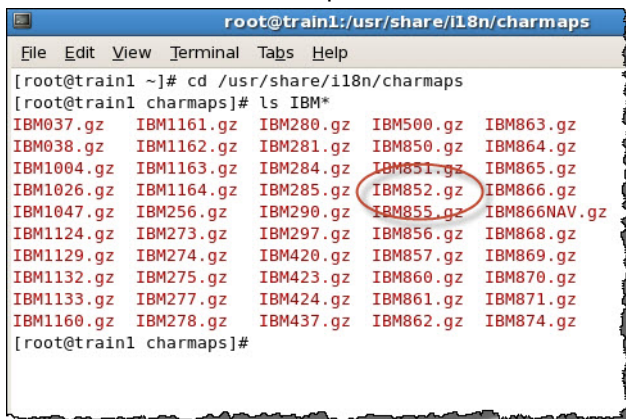
Add a text encoding plugin

Use this procedure to add support for a new encoding in a Genero Studio client-server configuration.

Before you begin, the preferred encoding must be installed on the Linux™ system.

The example shows how to configure a new encoding, using IBM852 as an example. The Genero Studio client runs on a Windows™ 7 Ultimate 64-bit platform using English with the CP1252 character set (Western European languages). The Genero Studio server runs on enterprise Linux™ using English with the UTF-8 character set.

1. Locate the IBM852 charmap on the internet or Linux™ system.



```

root@train1:~/usr/share/i18n/charmaps
File Edit View Terminal Tabs Help
[root@train1 ~]# cd /usr/share/i18n/charmaps
[root@train1 charmaps]# ls IBM*
IBM037.gz  IBM1161.gz  IBM280.gz  IBM500.gz  IBM863.gz
IBM038.gz  IBM1162.gz  IBM281.gz  IBM850.gz  IBM864.gz
IBM1004.gz  IBM1163.gz  IBM284.gz  IBM851.gz  IBM865.gz
IBM1026.gz  IBM1164.gz  IBM285.gz  IBM852.gz  IBM866.gz
IBM1047.gz  IBM256.gz  IBM290.gz  IBM855.gz  IBM866NAV.gz
IBM1124.gz  IBM273.gz  IBM297.gz  IBM856.gz  IBM868.gz
IBM1129.gz  IBM274.gz  IBM420.gz  IBM857.gz  IBM869.gz
IBM1132.gz  IBM275.gz  IBM423.gz  IBM860.gz  IBM870.gz
IBM1133.gz  IBM277.gz  IBM424.gz  IBM861.gz  IBM871.gz
IBM1160.gz  IBM278.gz  IBM437.gz  IBM862.gz  IBM874.gz
[root@train1 charmaps]#

```

Figure 109: Unix ls command showing the compressed IBM852 character map

2. Copy the uncompressed charmap to the `GSTDIR/conf/charmaps` directory on both client and server . Create the charmaps directory if it does not already exist.

```

root@train1:/usr/share/i18n/charmaps
File Edit View Terminal Tabs Help
[root@train1 charmaps]# gunzip IBM852
[root@train1 charmaps]# ls IBM*
IBM037.gz  IBM1161.gz  IBM280.gz  IBM500.gz  IBM863.gz  IBM875.gz
IBM038.gz  IBM1162.gz  IBM281.gz  IBM850.gz  IBM864.gz  IBM880.gz
IBM1004.gz IBM1163.gz  IBM284.gz  IBM851.gz  IBM865.gz  IBM891.gz
IBM1026.gz IBM1164.gz  IBM285.gz  IBM852     IBM866.gz  IBM903.gz
IBM1047.gz IBM256.gz   IBM290.gz  IBM855.gz  IBM866NAV.gz IBM904.gz
IBM1124.gz IBM273.gz   IBM297.gz  IBM856.gz  IBM868.gz  IBM905.gz
IBM1129.gz IBM274.gz   IBM420.gz  IBM857.gz  IBM869.gz  IBM918.gz
IBM1132.gz IBM275.gz   IBM423.gz  IBM860.gz  IBM870.gz  IBM922.gz
IBM1133.gz IBM277.gz   IBM424.gz  IBM861.gz  IBM871.gz
IBM1160.gz IBM278.gz   IBM437.gz  IBM862.gz  IBM874.gz
[root@train1 charmaps]# cp IBM852 /opt/fourjs/gst/gst/conf/charmaps
[root@train1 charmaps]#

```

Figure 110: Uncompressing and copying the charmap to the Genero Studio charmaps directory

3. Create a new entry for the encoding in `GSTDIR/conf/encodingMap.xml` on both client and server.

```

lisac@train1:/opt/fourjs/gst/gst/conf
File Edit View Terminal Tabs Help
<Alias name="949"          fallback="EUC-KR"          impl="EUC-KR"
  unixCountry="ko_KR" />
<Alias name="CP949"       fallback="EUC-KR,949"     impl="EUC-KR"
  unixCountry="ko_KR" />

<!-- UTF-8 -->
<!-- Genero support part of utf8 character set using the fglutf8 name -->
<Alias name="UTF-8"       fallback="utf8,fglutf8"   impl="utf8"
  unixCountry="en_US" />
<Alias name="utf8"        fallback="UTF-8,fglutf8"  impl="utf8"
  unixCountry="en_US" />
<Alias name="fglutf8"     fallback="utf8,UTF-8"     impl="utf8"
  unixCountry="en_US" />

<!-- Polish -->
<Alias name="IBM852"      fallback="852,ibm852"     impl="IBM852"
  unixCountry="pl_PL" />
<Alias name="852"         fallback="IBM852,ibm852"  impl="IBM852"
  unixCountry="pl_PL" />
<Alias name="CP852"       fallback="IBM852,852,ibm852" impl="IBM852"
  unixCountry="pl_PL" />

```

Figure 111: Adding the IBM852 alias to the encodingMap.xml file

4. Open Genero Studio on the client, and select **Tools >> Preferences >> General**. Check to see if the new plugin displays in the list of encodings.

If the plugin is not present in the Text Encoding list, check the following:

- Did you copy the charmap to the Genero Studio charmaps directory on both client and server?
- Is the charmap a valid POSIX2 charmap?

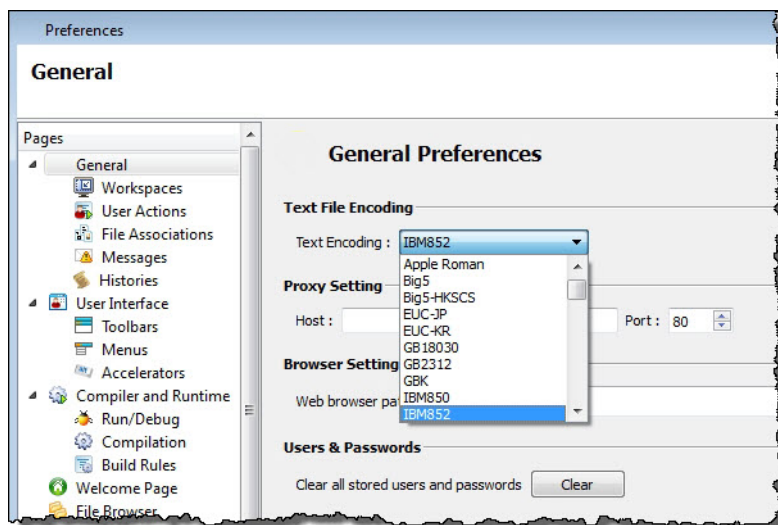


Figure 112: New text encoding plugin displayed in Text Encoding list.

Character mapping table (encodingMap.xml)

The `encodingMap.xml` file maps character set aliases, fallback character set names to try if the alias is unsupported, the name of the character set used by Genero Studio, and the corresponding language definition for the `LANG` variable for Genero on Unix.

A default list of text encodings is provided in Genero Studio. The mappings between each character set alias and its corresponding attributes are defined in the `encodingMap.xml` file in the `GSTDIR/conf` directory.

For some of the encodings there is only a partial match between Unix and Windows™ platforms.

This file contains a table used to map each encoding name (alias) to:

- A list of fallback character set names to try if the alias is not supported.
- The name of the character set used for encoding/decoding.
- An *implementation* name in Genero Studio (*impl* attribute)
- a `LANG` qualified country for Unix/Linux (*Country* attribute)

```
<Alias name="IBM852" fallback="852,ibm852" impl="IBM852"
  unixCountry="pl_PL" />
```

This means that:

- IBM852 is the name Genero Studio should use for this text encoding.
- 852 and `ibm852` are character set names to try if IBM852 is not supported.
- IBM852 is the name of the character set used for encoding/decoding.
- Under Unix or Linux™ the `LANG` language definition is `pl_PL`.

The *impl* attribute is defined within the POSIX2 charmap file after the `code_set_name`. It appears in the Genero Studio preferences combobox: `<code_set_name> IBM852`.

The *name* attribute is the name Genero Studio should use for this text encoding. By default it is the character set name (`code_set_name`); when an alias to an existing encoding is needed, the name attribute should contain the alias name.

```
<Alias name="CP852" fallback="IBM852,852" impl="IBM852" unixCountry="pl_PL" />
```

Configure keyboard and language on a Windows™ client

Follow this procedure to change the keyboard and default input language on a Windows™ client.

The details provided here are for Windows™ 7 Ultimate. Refer to the documentation for your version of Windows™ for assistance.

1. Open the **Clock, Language, and Region** dialog in the Control Panel and select **Region and Language**.
2. Select the **Keyboards and Languages** tab.
3. Select **Change keyboards...** and in the **Text Services and Input Languages** dialog, under **Default input language**, select the language you prefer as the default.

If the preferred language isn't listed, select **Add...** to add the language and preferred keyboard to the Default input language list.

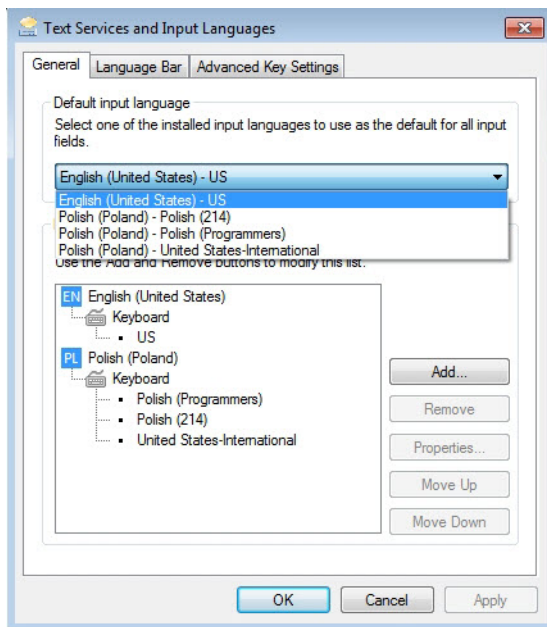


Figure 113: Text Services and Input Languages Dialog

Configure LANG on a Genero Studio Server

Follow this procedure to configure the locale on a Genero Studio server (Linux™).

This task assumes you have [added a remote host](#). In addition, the locale to be specified for the `LANG` environment variable must be installed on the Unix or Linux system.

Text encoding specifies the character set, and when compiling files, the `LANG` environment variable is used by Genero internal processes to determine which encoding the files use.

Note: Setting the `LANG` environment variable in an Environment Set overrides system defaults or user level (`.profile`) settings.

1. Launch the Genero Studio server (skip this step if using an `ssh` connection - it is automatic).
2. On the Genero Studio client, select the remote host from the list in the lower right corner.
3. On the Genero Studio client, navigate to **Tools >> Genero Configurations**.
4. Select a configuration from the list in the far left pane then select the plus



to add a new environment set.

5. Type in a meaningful name for the new environment set and press **OK**.

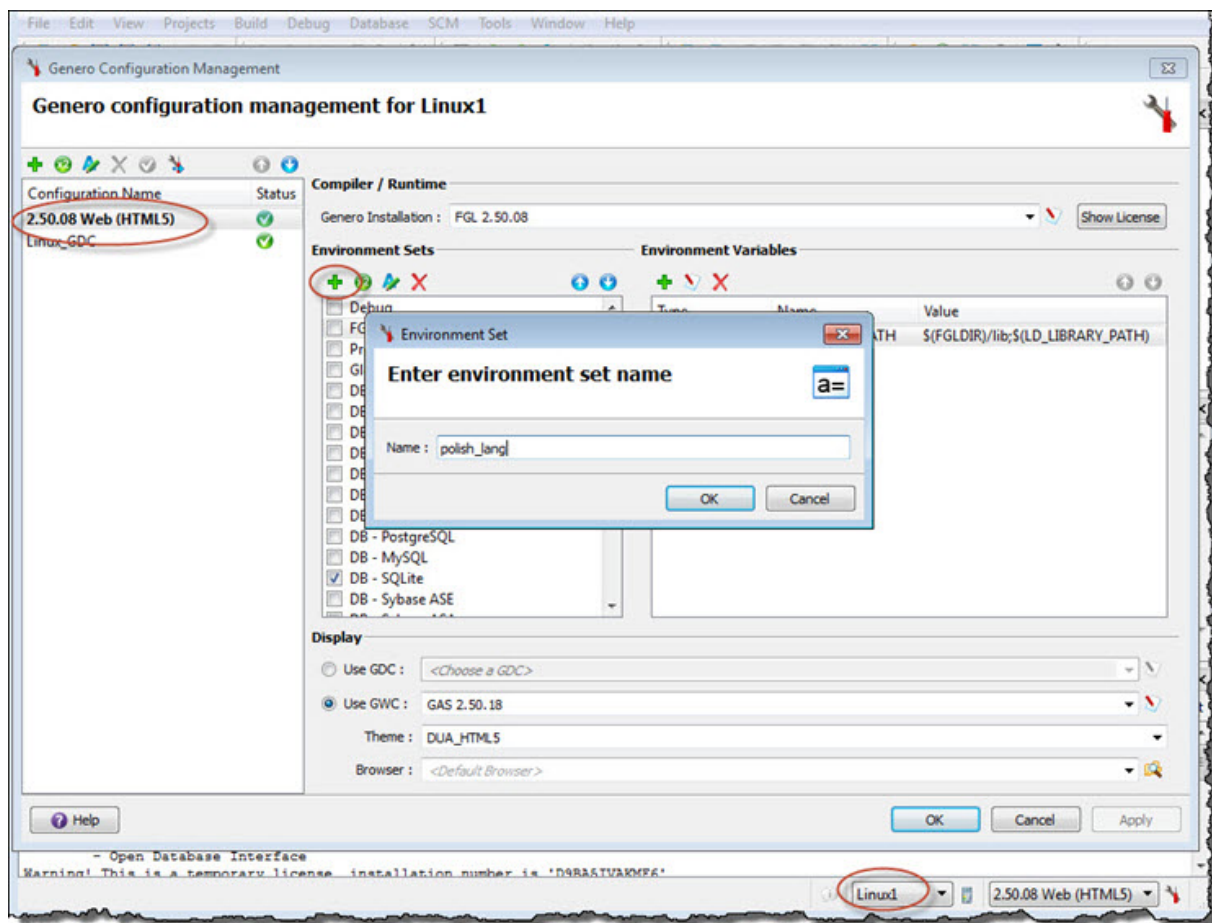


Figure 114: Add Remote Environment Set

6. Select the



above the Environment Variables list to add a variable named `LANG` with the desired locale (pl_PL.iso88592 in this example).

With the `LANG` environment variable, you define the language, the territory (country) and the character set (codeset) to be used. The format is:

```
<language>_<country>.<codeset>
```

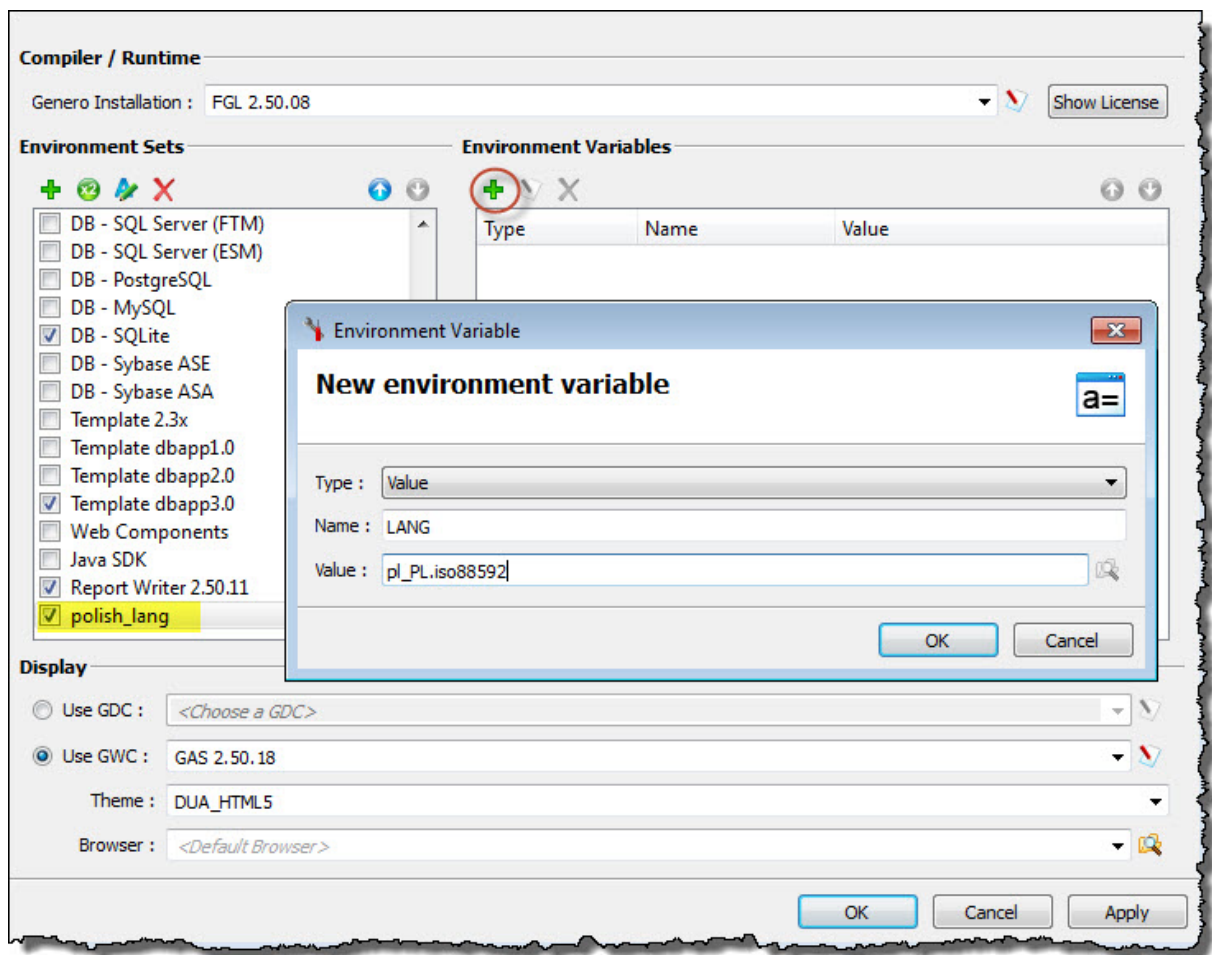


Figure 115: Configuring the LANG environment variable

7. Add the new environment set to other configurations for the remote host as needed.

Test text encoding configuration

Follow this procedure to test language settings in a Genero Studio client/server environment.

You can create a new project and compile it to test a new text encoding configuration or perform a simple test with the provided HelloWorld sample project as described below.

1. Select **Tutorials & Samples** in the Genero Studio Welcome Page and open the **HelloWorld** sample.
2. Double-click the `HelloForm.4fd` file to open it and scroll through the form properties until you locate the text property.
3. Use a translation tool such as Google Translate to create a phrase in the test language and replace the existing form text with the translated phrase.
4. Save the changes to the form, re-build and execute the **HelloWorld** application.

The language settings are correct if characters from the extended character set display properly.

Question marks displayed in place of extended characters indicate a problem with the configuration.

Configuration reference

Reference topics for configuring Genero Studio.

- [Genero Configuration Management dialog](#) on page 171

- [Import Configuration dialog](#) on page 173
- [Import Preferences dialog](#) on page 173
- [GAS standalone dispatcher: httpdispatch](#) on page 174

Genero Configuration Management dialog

Genero Studio configurations provide the information needed to run on your local or remote hosts.

A default configuration is provided, based on the software installed in the default directories. You can have multiple Genero configurations, distinguished by the name that you have assigned.

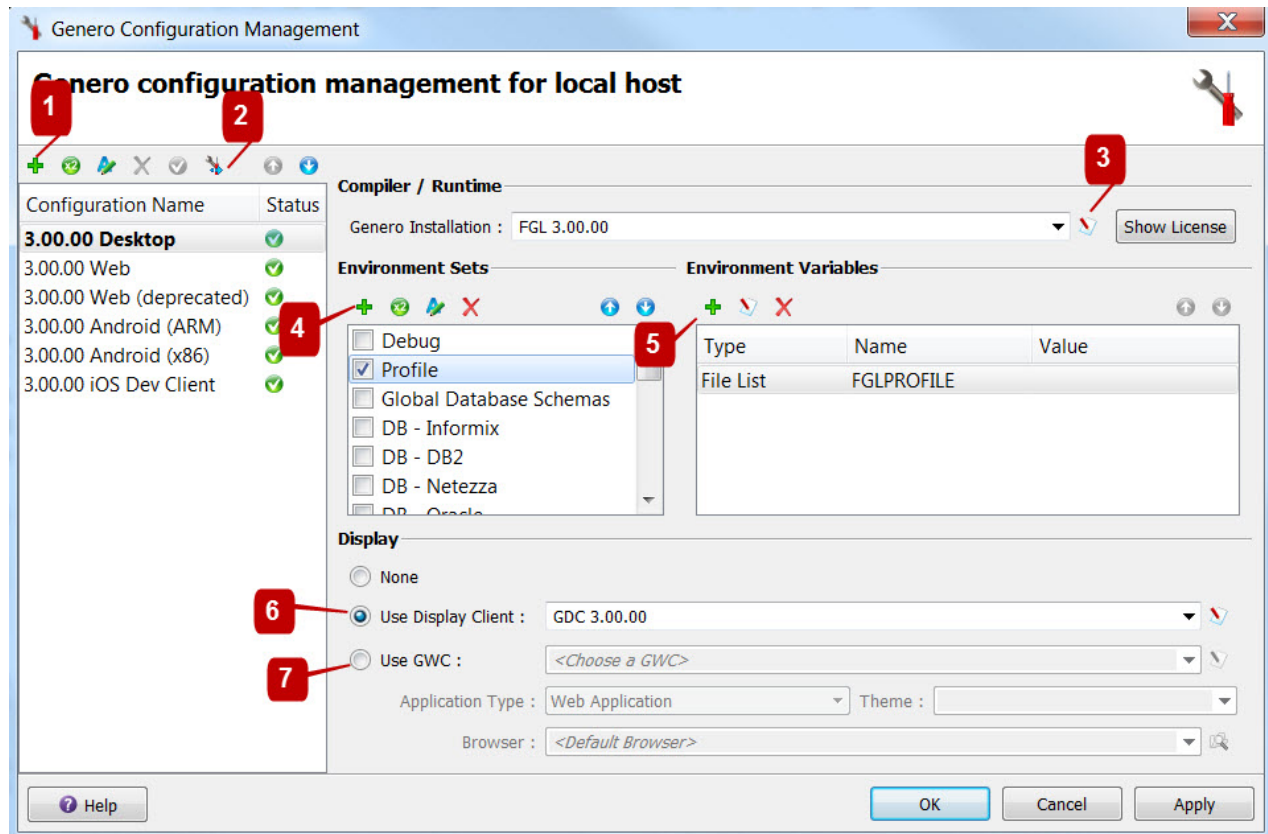


Figure 116: Genero Studio Configuration

1. Add a Configuration

The names of the available Genero configurations. Select the configuration name to view its settings. Use the integrated Toolbar to add the name of a new configuration. Once added, you can enter its settings.

A check mark in the **Status** column indicates there are no missing settings.

The integrated Toolbar also includes options to duplicate, rename, and delete configurations. Use the up or down arrow to modify the placement of the selected configuration in the list. The order of the configurations is organizational only and has no other effect.

2. Import a Configuration

Displays the [Import Configuration dialog](#) on page 173 to select an installation from which to import configurations.

3. Genero Installation

This field identifies which Genero Installation will be used to compile or run the application. A Genero Installation specifies the *FGLDIR* and the runner name. The names of the Genero installations that are available for the selected host are displayed in the list. Select the desired installation for your

configuration. To view or edit the settings for the selected Genero Installation name, or to add a new Genero Installation, select the **Edit** button (See [Genero Installations](#)).

4. Environment Sets

An environment set is a collection of environment variables to be set prior to compiling or running the application on the specified host. The name of the environment sets available for the selected host are listed. Use the up/down arrows to set the priority of environment sets. A check mark indicates the environment set is active. To view or edit the settings for the selected environment set, or to add a new environment set, select the **Edit** button. (See [Environment Sets](#)). Default environment sets have been created for the databases supported by Genero. Default variables have been entered for each set, and values have been provided where possible. Select your environment set, and enter any missing values or variables.

5. Environment Variables

Add and modify environment variables within an environment set by selecting the **Add** or **Edit** button.

6. Use Display Client

Possible front ends include:

- Genero Desktop Client (GDC)
- Android ARM
- Android x86
- iOS Development Client

[Desktop: GDC configurations](#) on page 146 and [Mobile clients: GMI and GMA configurations](#) on page 153 configurations can be viewed and edited by selecting the **Edit** button. (Genero Mobile only.)

7. Use GWC

[Web: GAS/GWC configurations](#) on page 147 configurations can be viewed and edited by selecting the **Edit** button.

When you select **Use GWC**, you can choose an available **Theme** to use when running an application. You can also override the web browser preferences setting and choose the web browser to use when running an application. If the theme field is left empty, then default theme will be used. If the web browser field is left empty, then the web browser defined in preferences will be used.

Confirm or cancel changes

Use these button options to confirm or cancel changes.

OK	Save and apply all modifications, then exit.
Cancel	Undo all modifications and exit. The last saved values are restored.
Apply	Confirms your updates, allowing you to test the new configuration.

Compiler / Runtime configuration (Genero Installations)

A Genero Installation contains information about the compiler and runtime version of Genero that will be used by Genero Studio.

Select the name of the **Genero Installation** in the Host list to view or edit the **Genero Installation settings**.

FGLDIR	The installation directory of the Genero BDL software.
FGL runner	The name of the Genero executable. (For example, <code>fglrun</code> , <code>fglrun.exe</code>)

FGL options

Options given to the FGL runner when an application is launched.

Select the **Check Installation** button to test the settings.

Confirm or cancel your changes.

Import Configuration dialog

When you install a new version of Genero Studio you can choose to import previous Configuration settings from older installations of Genero Studio.

The **Import Configuration** dialog can be displayed from the [Genero Configuration Management dialog](#) on page 171.

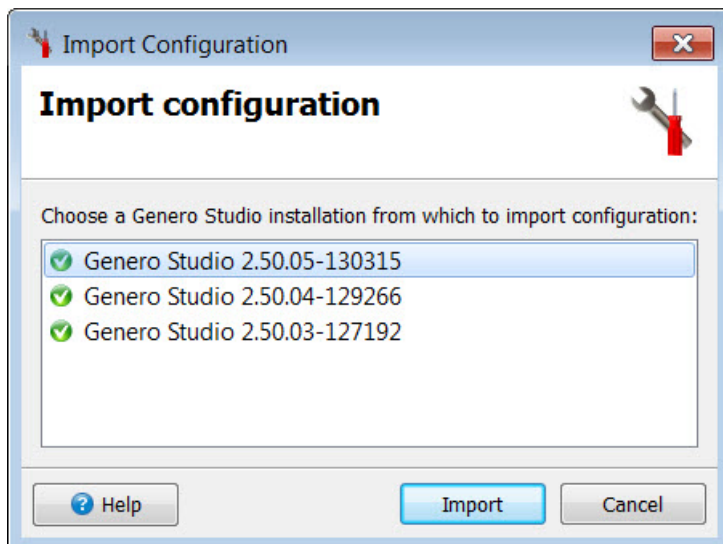


Figure 117: Import Configurations

Import Preferences dialog

When you install a new version of Genero Studio you can choose to import previous Configuration settings from older installations of Genero Studio.

The **Import Preferences** dialog appears when you install Genero Studio if prior installations are detected.

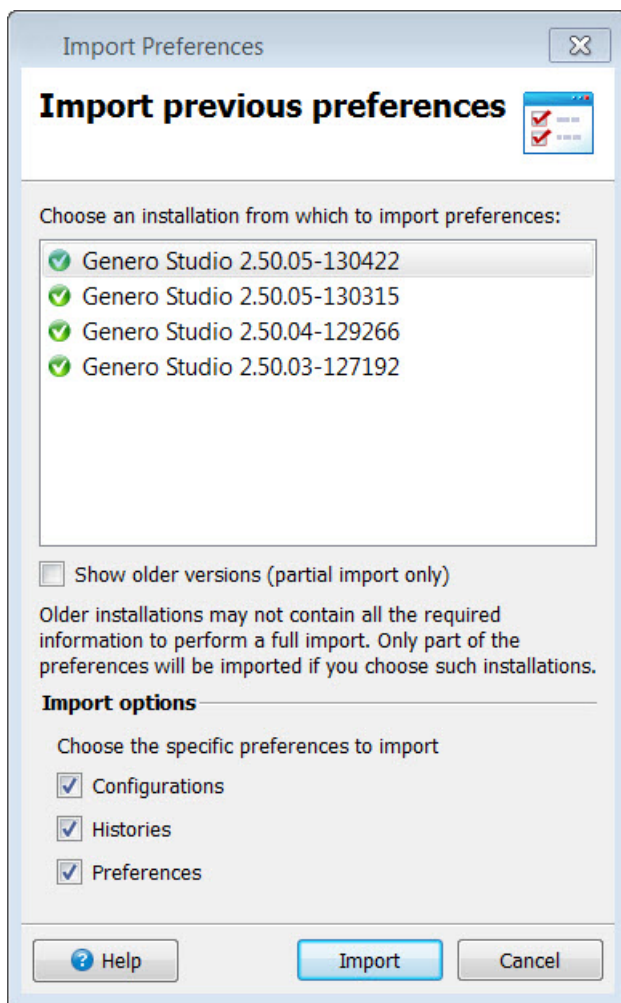


Figure 118: Import Preferences

Select an installation to import its Preferences settings and update the new installation of Genero Studio.

Show older versions

If the version of Genero Studio is too old, it won't be listed in the dialog unless the **Show old versions (partial import only)** checkbox is checked. If an old version is chosen, only part of the Preferences will be imported (for example, Preferences such as menus or Toolbars won't be imported).

Import options

Check the boxes for importing the **Configurations**, **Histories**, and/or **Preferences** from the selected installation.

GAS standalone dispatcher: httpdispatch

`httpdispatch` is the standalone dispatcher that starts GAS in command line. No web server is needed. It is only use in development mode. For deployment, use the dispatcher designed for your web server.

Launch `httpdispatch` at the command line by opening a Genero Workplace Window or by specifying it in your [GAS configuration](#) in Genero Studio.

See the topic **Standalone Genero Application Server** in the *Genero Application Server User Guide* for more information.

Syntax

```
httpdispatch [options]
```

Options**Table 23: httpdispatch options**

Option	Description
-h --help	Displays help information.
-p <i>directory</i> --as-directory <i>directory</i>	Specify the Genero Application Server directory.
-f <i>configuration_file</i> --configuration-file <i>configuration_file</i>	Specify which configuration file to use when starting the Genero Application Server dispatcher. If not specified, the default configuration file, <code>\$FGLASDIR/etc/as.xcf</code> , is used.
-k --no-keepalive	Disable keep alive for http connections. For debug purpose only.
-E <i>name=value</i> --resource-overwrite <i>name=value</i>	Overwrites the resource defined in the configuration file or creates a new one. Example: <pre>httpdispatch -E res.dvm.wa=\$FGLDIR/bin/myrun</pre> If in the configuration file "res.dvm.wa" has another value it is now set to myrun. The final value is the one set in the option .
-v --version	Displays version information.

Business Application Modeling (BAM)

Genero Studio Business Application Modeling (BAM) develops business applications from design diagrams rather than from writing code. It automatically generates the logic and source code for a database application to query, add, update and delete rows in database tables. BAM can generate desktop, web, and mobile applications.

- [Quick Start: Generate an application](#) on page 176
- [Quick Start: Generating a mobile app](#) on page 182
- [BAM Concepts](#) on page 192
- [BAM Projects](#) on page 200
- [Modeling the application](#) on page 202
- [Modeling the database](#) on page 226
- [Working with forms](#) on page 232
- [Adding custom code](#) on page 250
- [Modifying the look and feel](#) on page 264
- [BAM Reference](#) on page 268

Quick Start: Generate an application

This quick start guides you through generating a basic Genero business application, using the default template, that can be used to add, update, delete, and query rows in a relational database. BAM allows you to visually model your app and generates the code from the design models. You focus on the models, BAM handles the coding.

The application accesses the *officestore* sample database included in the Genero Studio distribution.

Genero Studio is already configured for the sample projects and databases. An `FGLPROFILE` file is in the `samples/DConfig` directory, and is used by the sample projects.

You may also create projects and applications using the sample databases and files.

Create a managed project

Before you begin, select **Tools >> Genero Configurations** and confirm that the **Template dbapp 3.0** or the template of your choice is selected in the Environment Sets list. This will be the code generation template used for the project.

1. Select **File >> New, Design, Managed Project (4pw)** to create a project for your generated application. When you create a managed project, nodes for the basic structure of the project are already defined and automatically contain the additional build rules needed for generated programs. The project structure has pre-defined nodes to contain the files for your project:
 - **Application_1** - for the program and form files, and any additional source code files
 - **Databases** - for the Genero database schema files (4dbx)
 - **Library_1** - for any additional resource files
2. Right-click on the Application node and select **Advanced Properties** from the menu. Note that the dependencies for the Databases and Library nodes have already been set, ensuring that any files they contain will be included in your application.
3. In the **Advanced Properties** dialog, select **Environment Variables** and add a User variable `FGLPROFILE` by clicking on the green cross. The `FGLPROFILE` file contains the configuration information to access the sample databases. This file is located in the `My Genero Files/samples/DConfig` directory.

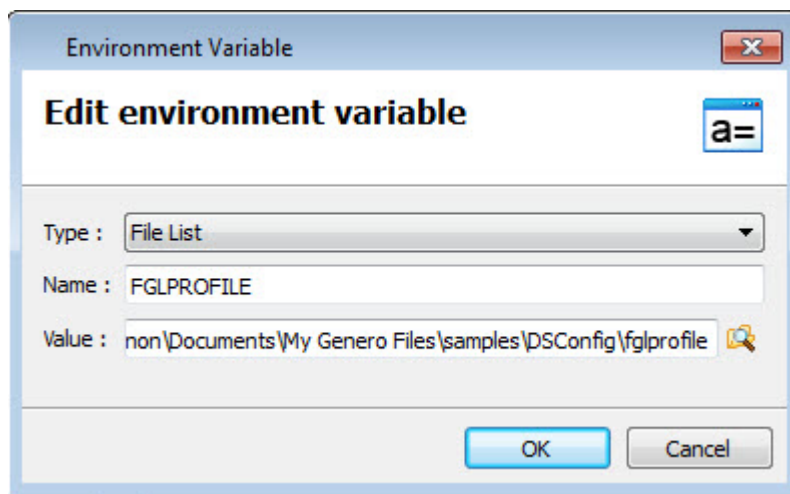


Figure 119: Edit environment variable dialog

4. Save the project (4pw) to a folder in your file system.

Add a meta-schema to the project (4dbx)

Your application will access the sample *officestore* database. The schema file for this database, **officestore.4dbx**, is located in `My Genero Files/samples/OfficeStore/database/`.

This 4dbx schema file is used to create items in your Business Application Diagram (4ba). It contains the information about the tables, columns, and relations of the relational database that is needed by your application.

1. Right-click the **Database** node in your project structure, and select **Add Files**. Locate the `officestore.4dbx` file and click the **Open** button to add the file to your project.
2. Save the changes to your project.

Create the Business Application diagram (4ba)

1. Select **File>>New, Design, Application Modeling, Business Application Diagram (4ba)** to create a blank diagram.
2. Right-click on the blank diagram and select **New >> Program**. This creates a Program entity on the diagram.
3. Right-click on the blank diagram and select **New >> CRUD Form**. This creates a Form entity on the diagram.
4. Right-click the Program entity, and select **New Relation**. Click and drag from the Program to the Form entity to create a relation from program to form.

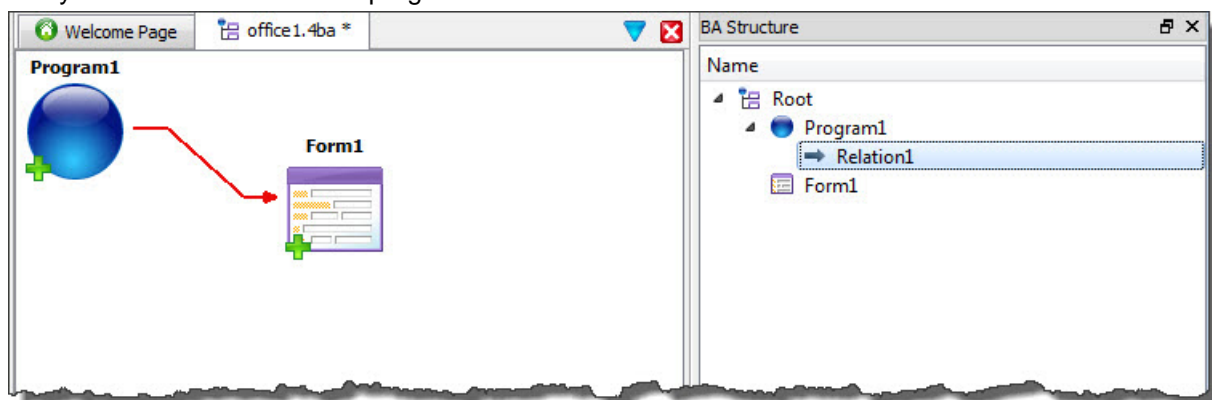


Figure 120: Defining a relationship

5. Save and name the BA Diagram (4ba) to your file structure and to the **Application** node in the project.

Implement the program and form

1. Right-click on the Program entity and select **Implement Program**. When the Save As dialog appears, assign a name for the program file, and save it under the Application node. This file will be used to generate the source code for the main function of the program. The file will have a 4prg extension.
2. Right-click on the Form entity and select **Implement CRUD Form from Database** to generate a form for the database table that you wish to access. Select the *officestore* database, the *account* table and the first ten fields of the table from the list. Transfer this selection to the Selected Fields list, which lists the desired fields for your form.

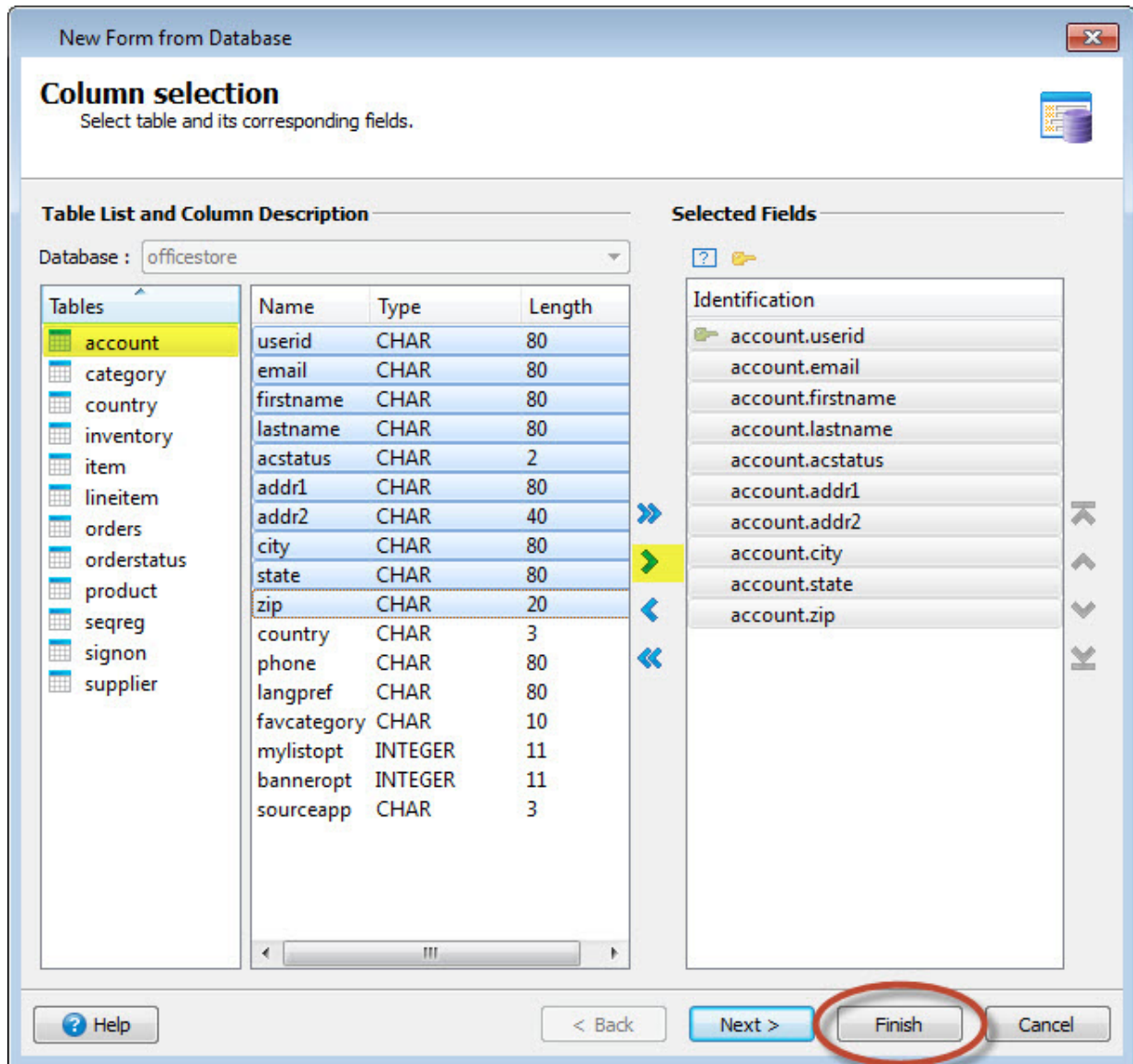


Figure 121: New Form from Database wizard Column selection

3. Click **Finish** and save the form (4fdm) to the project structure under the Application node.

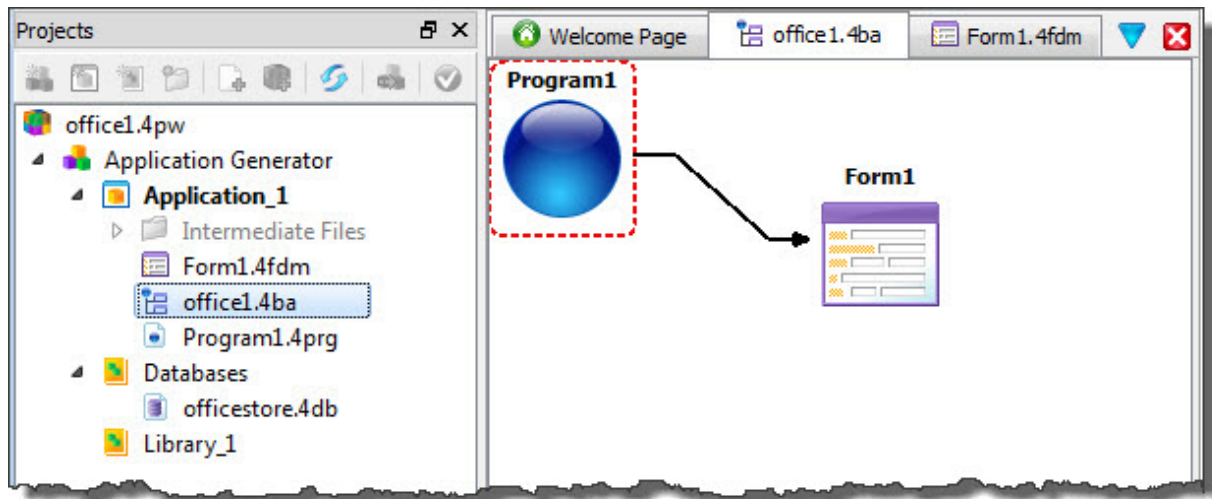


Figure 122: Projects Structure view

You now have two documents open; the Application Diagram (4ba) and the Form Definition (4fdm). The Project includes these new files.

4. Select **File>>Save All** to save the edited project and its contents.

Generate and run the application

1. To build a generated application, right-click the **Program** entity on the BA diagram (4ba) and select **Build Program** from the context menu. Alternatively, right-click the Application node in the project and select **Build**. The results of the build are displayed in the Output view.

To better understand what is happening during the build of the program, turn on verbose mode using **Tools >> Preferences, Compiler and Runtime, Compilation Configuration**.

The generated and compiled files will be stored in the path specified in the Target Directory property of the Group node in the project.

2. To execute a generated application, right-click the **Program** entity in the BA Diagram (4ba) and select **Execute Program**. Alternatively, right-click the Application node in the project and select **Execute**. Toolbar icons for a program's default actions will be generated, depending on the properties selected for the module's screen records and the fields on the form. The necessary business logic was created in the generated BDL files (4gl) to implement the relevant actions. See [Working with forms](#) on page 232.

User Id	<input type="text" value="smith"/>	E-mail	<input type="text" value="charles@smith.com"/>
First Name	<input type="text" value="Charles"/>	Last Name	<input type="text" value="Smith"/>
Status	<input type="text" value="OK"/>	Address	<input type="text" value="49 Roche Way"/>
Address	<input type="text"/>	City	<input type="text" value="Columbus"/>
State	<input type="text" value="OH"/>	Zip code	<input type="text" value="75038"/>

1/6 OVR

Figure 123: A generated application running on the desktop (GDC)

Add a detail list to the form

A Genero application can display a form that contains a master-detail relationship between two tables. The user can search for a row in the master table, and the corresponding rows in the detail table will also be displayed. The values in rows from both tables can be added, deleted, or modified.

The form must contain fields from both tables and the table relationships must be set.

1. In the form, enlarge the grid around the master table to make room for the fields from the other table.
2. Select **Container >> Data Control** and select these columns from the orders table: orderid, userid, orderdate, totalprice and creditcard. The container for these fields can be a table (to display multiple rows). Click **Next**, select a **Table** container, click **Finish** and draw the container within the enlarged grid, under the original.

This form design now contains fields from the master table, **account**, and a detail table, **orders**.

The screenshot shows a form design tool interface with two tabs: 'Welcome Page' and 'Form1.4fdm'. The form contains two main sections. The top section is a form with input fields for user information, organized into two columns. The left column includes fields for 'User Id', 'First Name', 'Status', 'Address', and 'State'. The right column includes fields for 'E-mail', 'Last Name', 'Address', 'City', and 'Zip code'. Below this is a table with five columns: 'Order Id', 'User Id', 'Order Date', 'Total Price', and 'Credit Card Number'. Each cell in the table contains the word 'Edit', indicating that the table is interactive and allows for editing data. The table has five rows, and the first row is highlighted.

Figure 124: Form tab

3. Select the **Records** tab. Select the master table record and make sure the **active** property is checked. Select and identify a column as the unique key, if not already identified, as indicated with the key icon.

The screenshot shows the 'Records' tab in the design tool. On the left, a list of records is displayed under the heading 'Record1 (Master)'. The first record is selected and has a key icon next to it, indicating it is the unique key. The fields listed are: account.userid, account.email, account.firstname, account.lastname, account.acstatus, account.addr1, account.addr2, account.city, account.state, and account.zip. On the right, the 'Properties' window is open, showing the properties for the selected record. The 'active' property is checked, and the 'masterTable' property is set to 'account'.

Name	Value
Object	Record1
Managed	
active	<input checked="" type="checkbox"/>
masterTable	account
Query	
Functionality	

Figure 125: Master table record

4. Do the same for the **detail table record**.

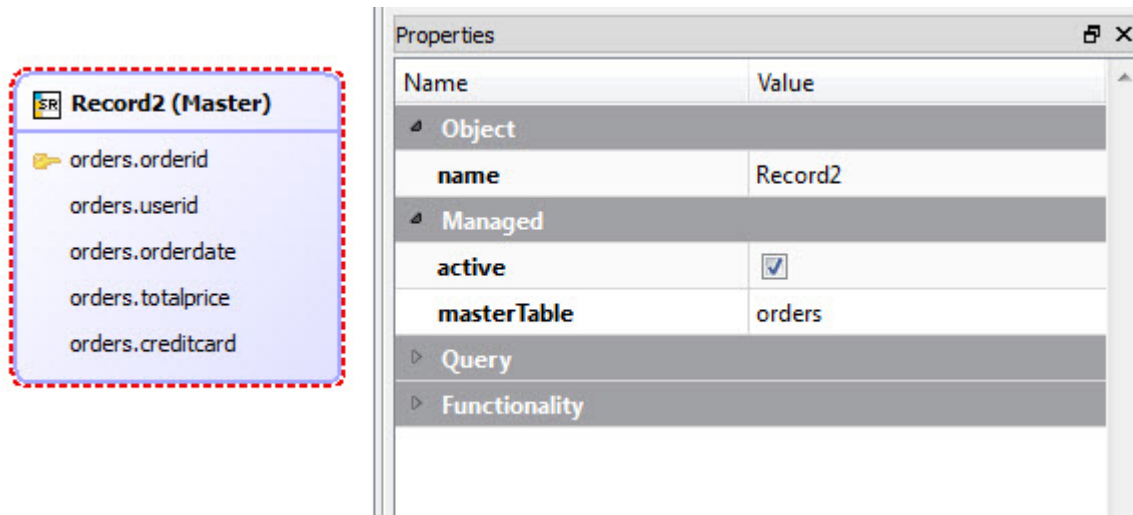


Figure 126: Detail table record

5. Set the relationship between the master and detail records. Right-click the foreign key field in the detail table record (**orders.userid**, in our example), and select **Add Relation To**. Drag the arrow to point to the primary key in the master table record (**account.userid**).

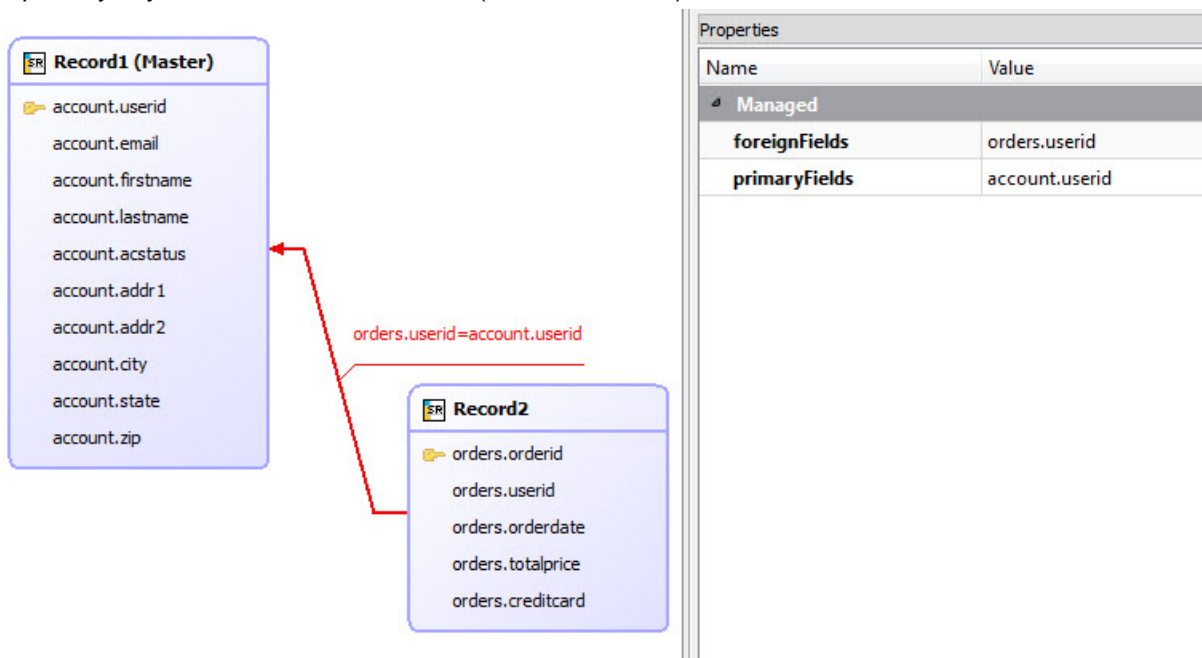


Figure 127: Master-detail relationship

6. Select **File>>Save All** to save the project and the modifications to its contents.
7. Build and execute the program.

The screenshot shows a web browser window titled 'Form1'. The browser's address bar contains 'File Edit Navigate'. The toolbar includes icons for Accept, Cancel, New, Append, Insert, Modify, Delete, Search, and navigation buttons (First, Previous, Next, Last). The form contains the following fields:

User Id	miller	E-mail	steve@miller.com	
First Name	Steve	Last Name	Miller	
Status	OK	Address	191 North Orange Street	
Address		City	Orange	
State	CA	Zip code	92866	

Below the form is a table with the following data:

Order Id	User Id	Order Date	Total Price	Credit Card Number
2	miller	10/25/2011	1634.85	

The status bar at the bottom of the browser shows '3/6' and 'OVR'.

Figure 128: The application executes

Quick Start: Generating a mobile app

This quick start guides you through generating and running a simple, one form mobile app. The Business Application Modeler (BAM) allows you to visually model your app and generates the code from the design models. You focus on the models, BAM handles the coding.

Important: This quick start assumes you have configured Genero Mobile to run an app to a mobile device or emulator. See the appropriate configuration topics.

- [Create a BAM mobile project](#) on page 182
- [Create a database](#) on page 184
- [Create form from database](#) on page 187
- [Generate and run the app](#) on page 187
- [Add phone functionality to the app](#) on page 188
- [Customize the app](#) on page 190
- [Package and Deploy](#) on page 191

Create a BAM mobile project

Create a new mobile project. By using the structure imposed by this project type, building, packaging and deploying your app to a mobile device is simplified.

1. Select **File >> New, Design, BAM Mobile Project (4pw)** to create a project for your generated app.
2. Identify a project name and directory location for your project files.

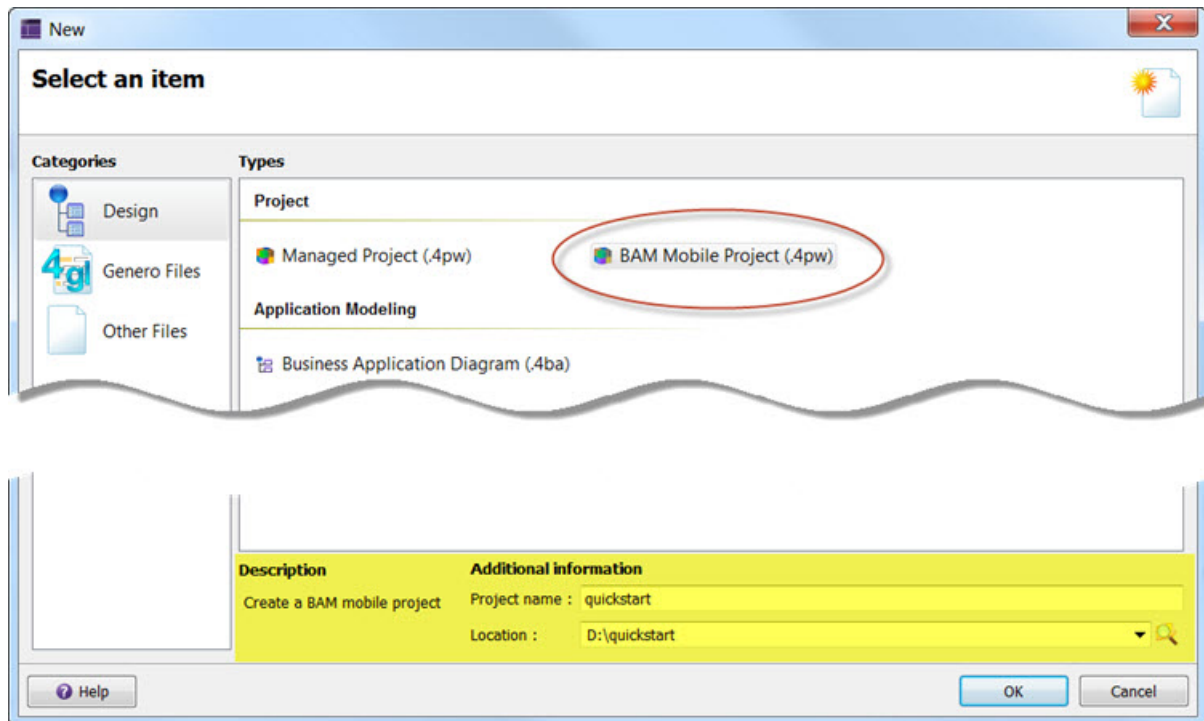


Figure 129: BAM Mobile Project

- From your new project, expand the **Project** group. Right-click on the **Application** node and select **Advanced Properties**. Note that the dependencies have already been set, ensuring that any files contained in the checked nodes will be included in your app.
- Launch your connected emulator. This quick start assumes you have configured Genero Mobile to run an app to a mobile device or emulator. See *Configuring Genero Mobile for development* in the *Genero Mobile Developer Guide* to set up your iOS or Android mobile device or emulator.

Mobile projects

BAM Mobile Projects have predefined nodes for the basic structure of the project. These nodes correspond to the project directories created on disk in the project directory you specified. The project also includes the build rules needed to generate, package, and deploy the application.

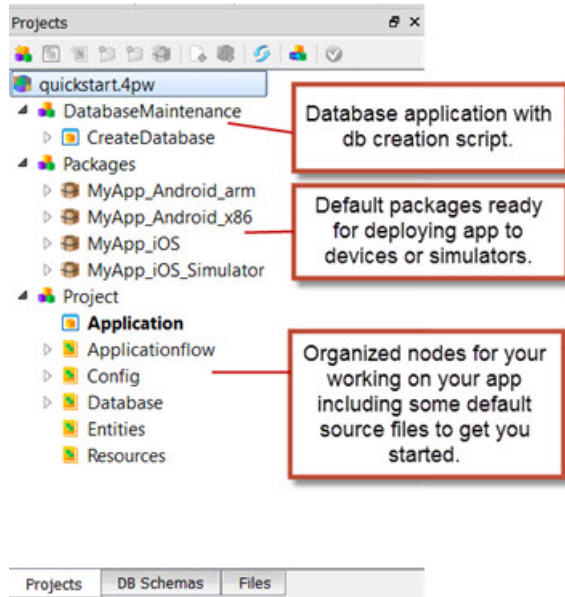


Figure 130: Default Project structure

Table 24: Default Project structure

Node	Files to be saved to this node
Application	4prg (implemented program file from 4ba diagram)
Applicationflow	4ba
Config	FGLPROFILE
Database	4dbx, db
Entities	4fdm (forms or other entities from 4ba diagram)
Resources	XML files such as 4st for styles

Create a database

Create a new database for your project.

There is a 4dbx file already in the project in the **Database** node. The 4dbx contains the information about the tables, columns, and relations of the relational database that is needed by your app.

1. In the **Project** group, expand the **Database** node, and double-click on the 4dbx to open it. The Meta-Schema Manager is launched for viewing and modifying the schema.
2. Right-click and select **Add Table**. Set the table's name property to `account`. Right-click on the table to add the columns as shown.

+ account			
Name	Type	NN	
+ id	SERIAL	<input checked="" type="checkbox"/>	
+ firstname	CHAR(32)	<input type="checkbox"/>	
+ lastname	CHAR(32)	<input type="checkbox"/>	
+ company	CHAR(32)	<input type="checkbox"/>	
+ phone	CHAR(32)	<input type="checkbox"/>	

Figure 131: Create table

3. Make the `id` column `SERIAL` and **not null**. Make the `id` column the primary key for the table. With the table selected, right-click to see the menu option for adding a constraint.

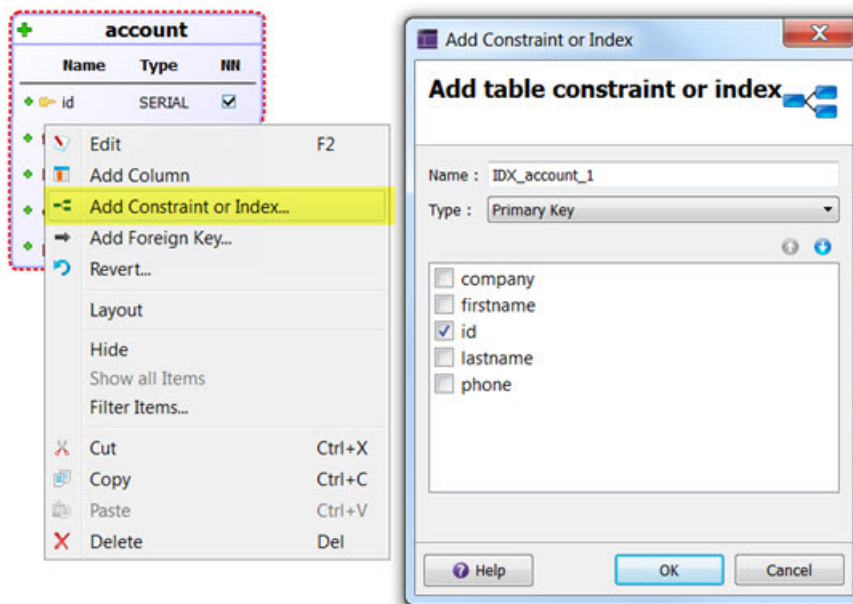


Figure 132: Set primary key

4. Save your file.
5. Right-click on the `4dbx` file in your project and select **Generate Database Creation Script**. Select the option to populate database with sample data and then **Generate**.

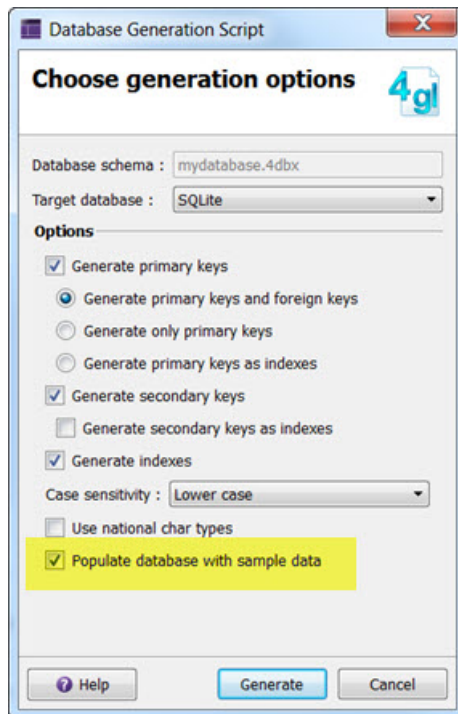


Figure 133: Generate Database Creation Script

6. Save the file to the `databaseMaintenance` sub directory in your project directory to *overwrite* the default file with your file.
 - a) Uncheck the **Insert the file in the project** option as the file is already in the project; you are just saving the file to disk.

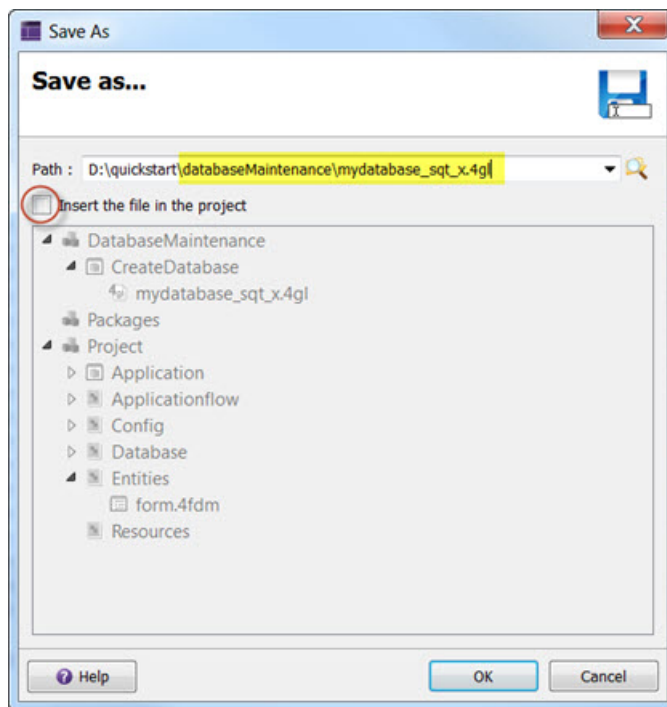


Figure 134: Save to databaseMaintenance sub directory

7. In the project, expand the **DatabaseMaintenance** group. Right-click on the **CreateDatabase** program and select **Execute**. This will compile and run your script, creating and populating the database.

Create form from database

Create a form for your project.

1. Expand the **Applicationflow** node and open the `appflow.4ba` file. This is your Business Application Diagram and shows the flow of your program. You will see there is one Program entity (**main**) and one Form (**form**) entity.
2. Right-click on the Form entity and select **Implement CRUD Form from Database** to generate a form for the database table that you wish to access. Select all the fields in your table and transfer this selection to the Selected Fields list, which lists the desired fields for your form. Select **Finish** to accept the defaults from the rest of the wizard prompts.
Per the default, this form will be organized in a Grid layout.
3. Save your form to your physical project directory (for example `D:\quickstart`) and in the **Entities** node of the project.
4. Return to the `appflow.4ba` file. Right-click on the Program entity (**main**) and select **Implement Program**.
5. Save the program to your physical project directory (for example `D:\quickstart`) and in the **Application** program node.
6. Save the changes to your project.

Generate and run the app

Run your app to display to a device or emulator.

1. Right-click on the **Application** node in the project or on the **Program** entity in the BA diagram and select **Build**.

Note: To better understand what is happening during the build of the program, turn on verbose mode using **Tools >> Preferences, Compiler and Runtime, Compilation Configuration**

2. Right-click on the **Application** node in the project or on the **Program** entity in the BA diagram and select **Execute**.

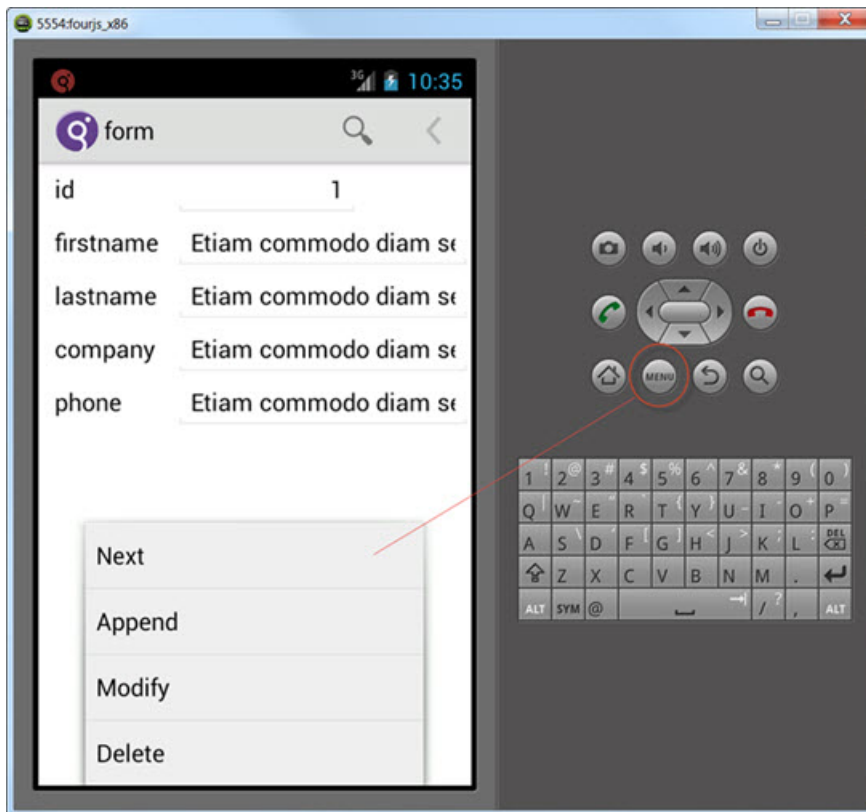
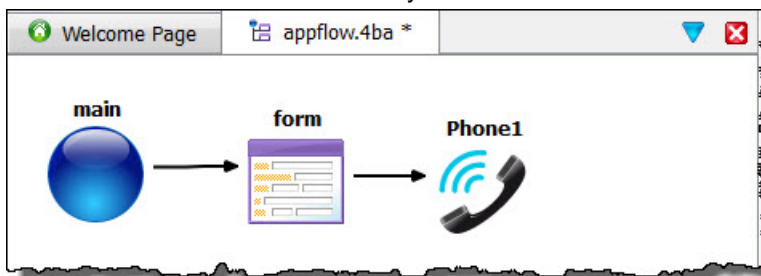


Figure 135: First app running on Android emulator

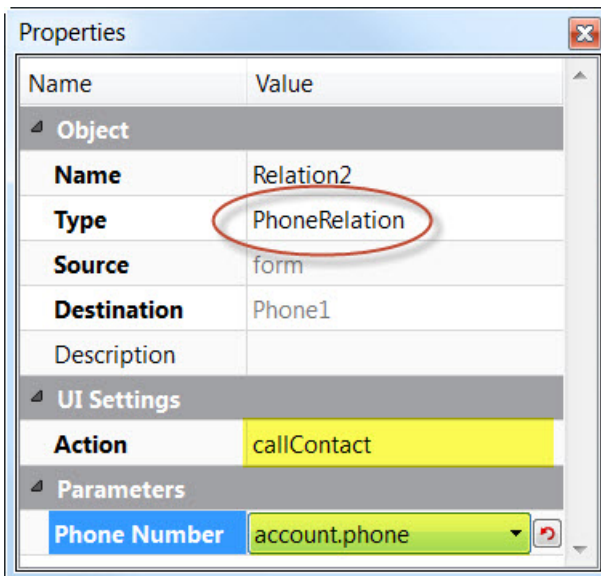
Add phone functionality to the app

You can generate the code to interact with the mobile device's features. This example shows how to call a selected phone number from the app.

1. Open the `appflow.4ba` file.
2. Right-click on the BA diagram and select **New >> Phone**.
3. Right-click on the **Form** entity and select **New Relation**. Click on the **Form** entity again and drag the **Relation** arrow to the **Phone** entity.

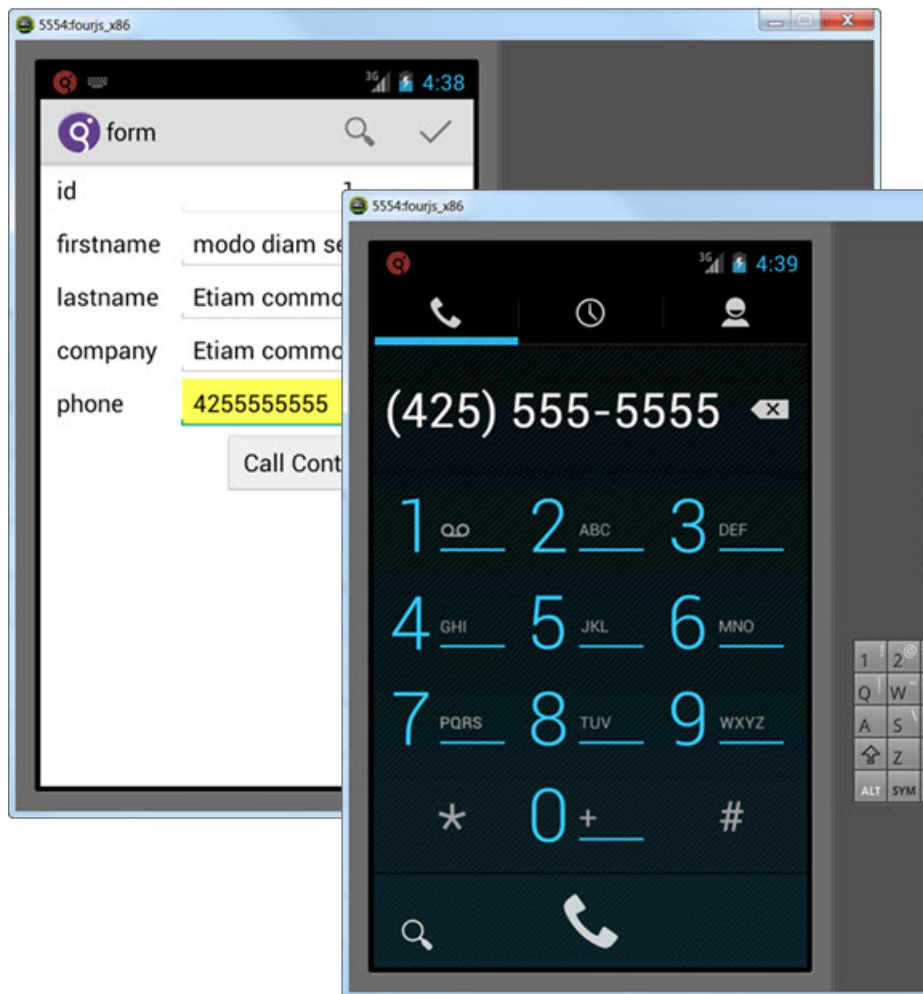


4. Select the **Relation** between the **Form** and the **Phone** entity and set the **Action** property to `callContact`. Set the **Phone Number** property to the phone column (`account.phone` in our example database).



5. Save your changes to `appflow.4ba`.
6. Double-click on the **Form** entity to open it in Form Designer. Add a button that triggers your `callContact` action.
 - a) Expand the canvas and the grid that is containing the form fields if you need room to add a button.
 - b) Select **Widget >> Button** from the menu.
 - c) Draw a button onto your form. Set the `name` property of your button to `callContact`. The name of the button is also the action to be triggered by the button.
 - d) Set the `text` property of the button to `Call Contact`.
7. To preview your form, select **Build >> Preview**.
8. Make further changes to your form if you wish.
9. Save your changes to the form.
10. Execute your program, this will rebuild your program and execute it in one step.
11. Test the new button.

Tip: The sample data in the phone field is not a valid phone number. Run the app and select **Modify** to update the phone field with a phone number of your choice. On the Android emulator, the **MENU** button displays the available program actions.



Customize the app

Customize the app by changing a property value in the BA diagram.

At this point, you would continue to build your BA diagram to add and customize forms, zoom forms, and mobile peripheral entities, relations between the entities, and to customize the generated code as needed. There are many customization options in BAM.

Table 25: Customization example resources

Example	Resource
Creating forms for mobile devices.	See <i>Working with Forms</i> section in the <i>Genero Studio User Guide</i> .
Mobile form patterns. The Mobile Patterns demo includes a BA diagram with examples of various forms, relationships between forms, and form behavior.	See the <i>Genero Mobile demo applications</i> section in <i>Genero Mobile Guide</i> and the <i>Mobile form patterns</i> topic in the <i>Genero Studio User Guide</i> .
Add custom code to the BAM generated code.	See the <i>Adding custom code</i> topic in the <i>Genero Studio User Guide</i> .
Change default rendering of the actions in the app.	See the <i>Action rendering</i> topic in the <i>Genero Mobile Developer Guide</i> .

For the purposes of this quick start, make a simple customization to your app by changing the form to open in **ADD** mode, instead of the default **DISPLAY** mode.

1. Open `appflow.4ba`.
2. Select the **Form** entity. The properties view shows all of the properties set on the form. Find the **UI Settings** section. (You can always display a view with the menu **Window >> Views**).
3. Change the `Open Mode` property to **ADD**.

This changes the program so that this form will open ready for the user to enter a new row of data.

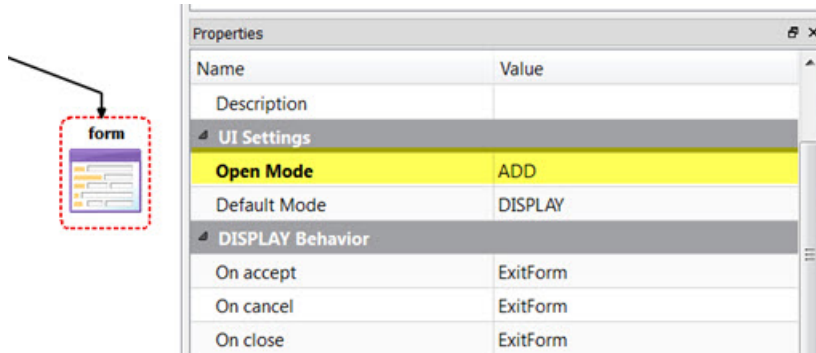


Figure 136: Change a form property

4. Save your changes to `appflow.4ba`.
5. Execute the app again. (Right-click on the program entity (main) and select **Execute**.) New code is generated to reflect this property change.
6. Test the program by adding a new row.

Package and Deploy

Package the app and deploy it for testing on your connected device or emulator.

1. Expand the **Packages** group in your project.
2. Find the package that corresponds to the device or emulator you have configured.



Figure 137: Pre-configured package types

3. Right-click on the package and select **Deploy**. This will build the package and deploy it.

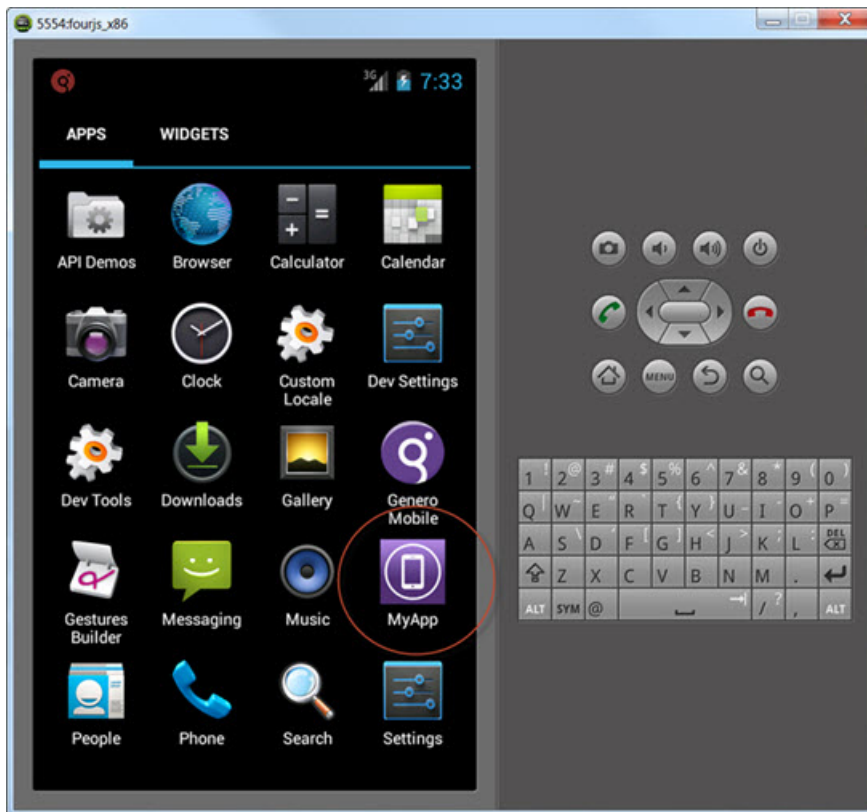


Figure 138: App deployed to Android emulator

4. Tap to launch the app.

BAM Concepts

This section includes overview information about BAM.

- [What is Business Application Modeling \(BAM\)?](#) on page 192
- [How code is generated](#) on page 194
- [The modeling diagrams](#) on page 196
- [Mobile apps vs Desktop applications](#) on page 198
- [The default template features](#) on page 199

What is Business Application Modeling (BAM)?

Genero Studio Business Application Modeling (BAM) develops business applications from design diagrams rather than from writing code. It automatically generates the logic and source code for a database application to query, add, update and delete rows in database tables. BAM can generate desktop, web, and mobile applications.

BAM design process

BAM provides a complete framework for design and development, incorporating the entire life cycle of an application.

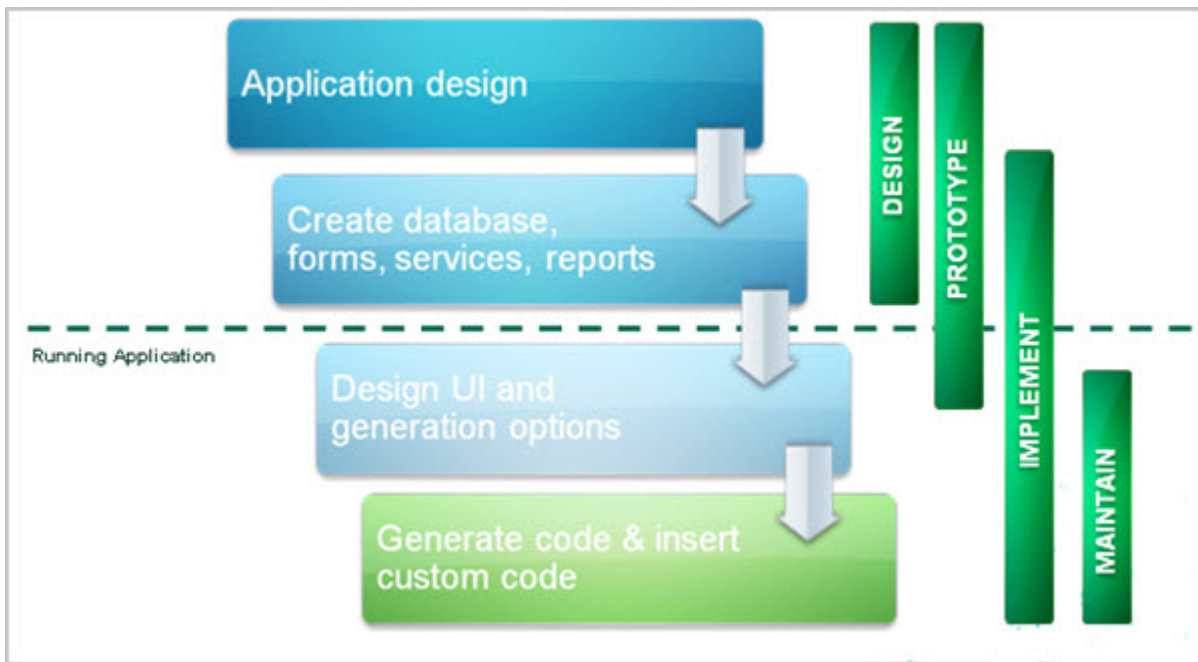


Figure 139: Application life cycle

With BAM:

- Create a project type pre-configured for a desktop or mobile app.
- Model the entities of your applications and their interconnection in a high level business diagram.
- Model the detailed design of your database, forms, services, and reports in sub-diagrams.
- Select options for the entities for user interface design and code generation.
- Automatically generate an application matching the behavior described in the diagrams without coding.
- Enhance the generated code with your own logic.

Is my application a good candidate for BAM?

All CRUD forms -- to view, add, update, or delete data from the database are perfect candidates for BAM.

BAM does not generate code for:

- Non-database forms, such as login screens or forms displaying only computed fields
- Web components
- Specific process logic, such as database synchronization for mobile apps

Custom code can be added, however, for any functionality that is not generated by BAM.

How code is generated

When you build an application from a Business Application diagram, the build rules define the various files that are input into the Code Generation Engine and the application code files that are output.

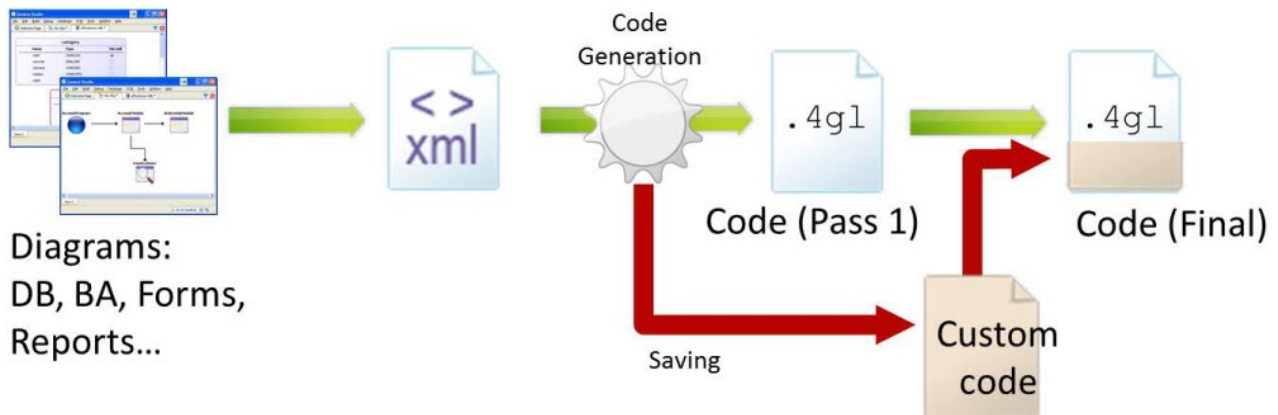


Figure 140: Code Generation flow

BAM Consolidates and Generates

Data Consolidation

The input from the BA diagram and related entities is gathered into a single XML file, which consolidates all the inputs into one package. This file is used when processing and generating the application code. This file could also be used to provide input to create the application models.

Code Generation

The XML file and a code template are used to generate the application code. The default Tcl template produces Genero 4g1 files, but another tool could be used to generate the code (XSL translator for example). Custom code is preserved; if any custom code was created earlier, it is automatically restored in the newly generated application code.

Example

The build rules define the series of commands used to build and generate the code. In general, the Build rule for code generation:

- Saves custom code added by the user to the generated source files
- Generates the new source files without user code
- Restores the user code in the generated files
- Compiles the written and generated source files
- Links the compiled files

View default build rules selecting **Projects >> Edit Build Rules**. This example shows the build rules used to generate the code for a Program entity in the Business Application diagram.

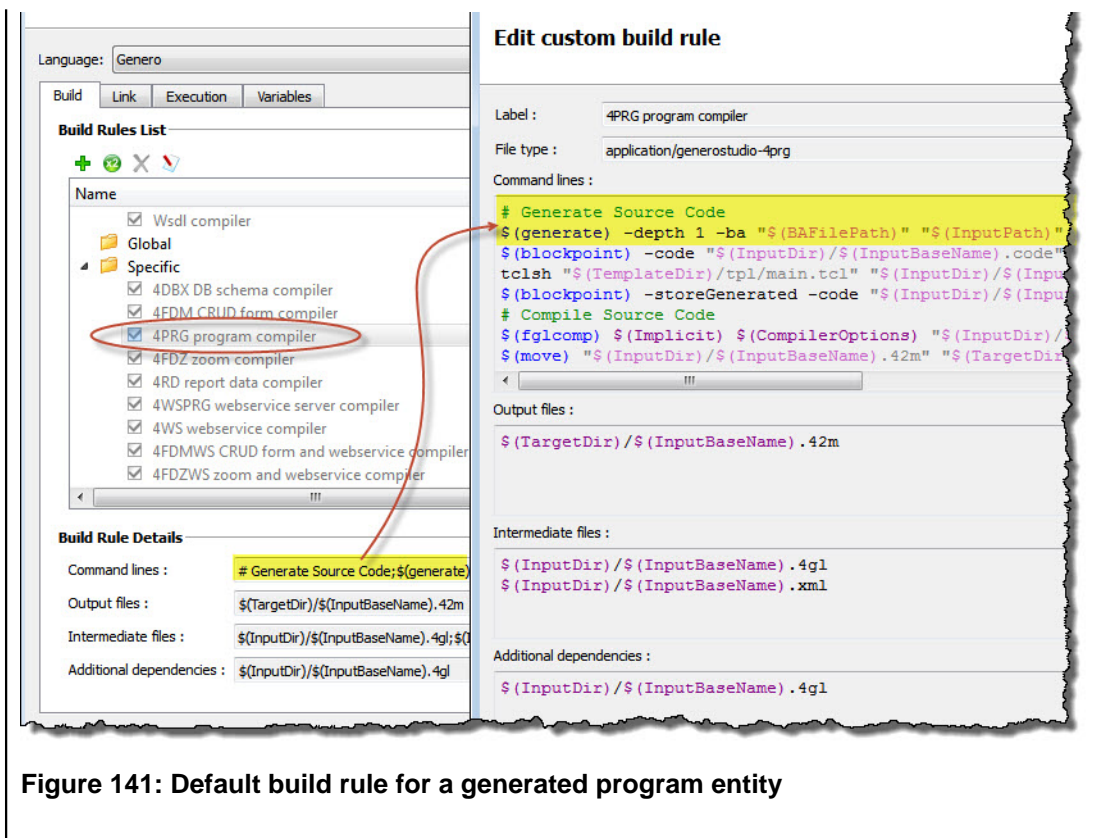


Figure 141: Default build rule for a generated program entity

Table 26: Default build rule example

Build rule command	Description
<code>\$(generate)</code>	The <code>\$(generate)</code> command creates an intermediary XML file from modeled entities.
<code>\$(blockpoint) -code</code>	BLOCK/POINT is extracted from previously generated and modified code.
<code>tclsh</code> on page 269	The <code>tclsh</code> executable generates the final file by using both a Tcl template file and the intermediary XML file created by the <code>\$(generate)</code> command.
<code>\$(blockpoint) -storeGenerated</code>	Extracted BLOCK/POINT code is put back into the generated code.
<code>\$(fglcomp)</code>	The <code>fglcomp</code> tool compiles BDL program sources files into a p-code version.
<code>\$(move)</code>	Moves the given file or directory to the given destination in a platform independent way.

Reviewing the Build

To better understand what is happening during the build of the program, turn on verbose mode using **Tools >> Preferences, Compiler and Runtime, Compilation Configuration**. Compile a diagram file, program, or application and view the results in the output.

The modeling diagrams

Business Application Modeling is based on several diagrams. Each diagram has its own purpose for modeling the application's features and behavior.

Diagrams are used at all stages in the development. They serve as inputs for code generation, and are considered source code for the application. They are always up to date. Diagrams can be customized with additional properties using the [settings.agconf](#) configuration file.

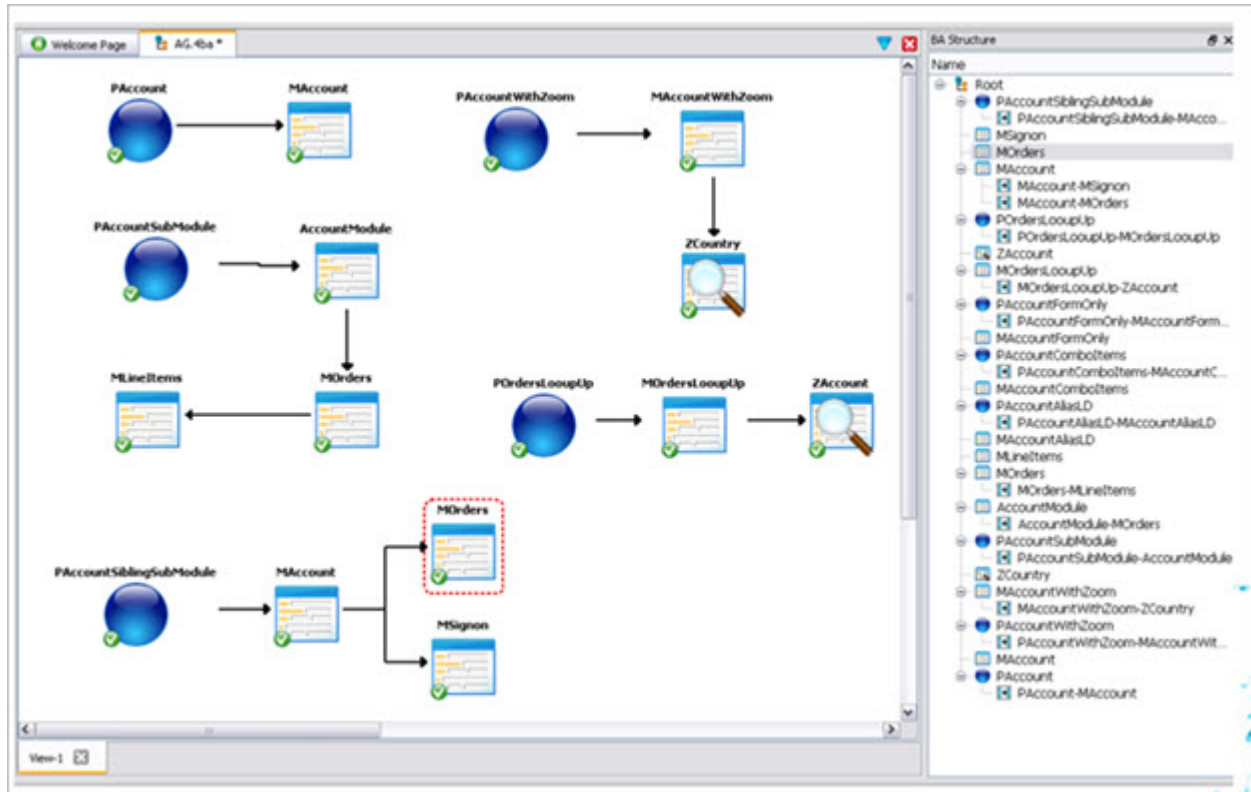


Figure 142: Business Application diagram

The [Business Application diagram](#) models the application flow and provides a high-level overview of the application.

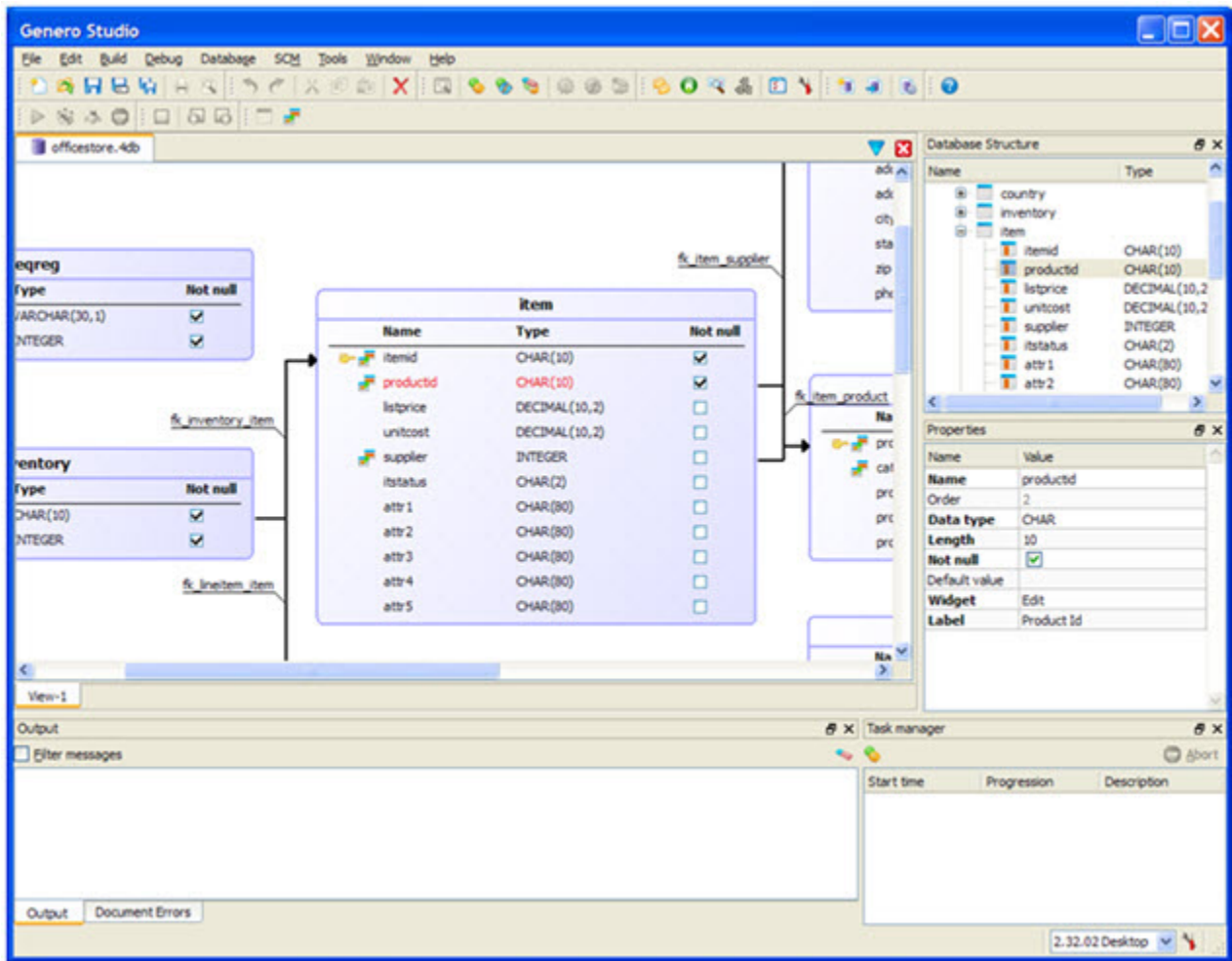


Figure 143: Meta-schema Manager diagram

The Meta-schema Manager diagram models the database tables, columns, and constraints. See [Meta-schema Manager](#) on page 288.

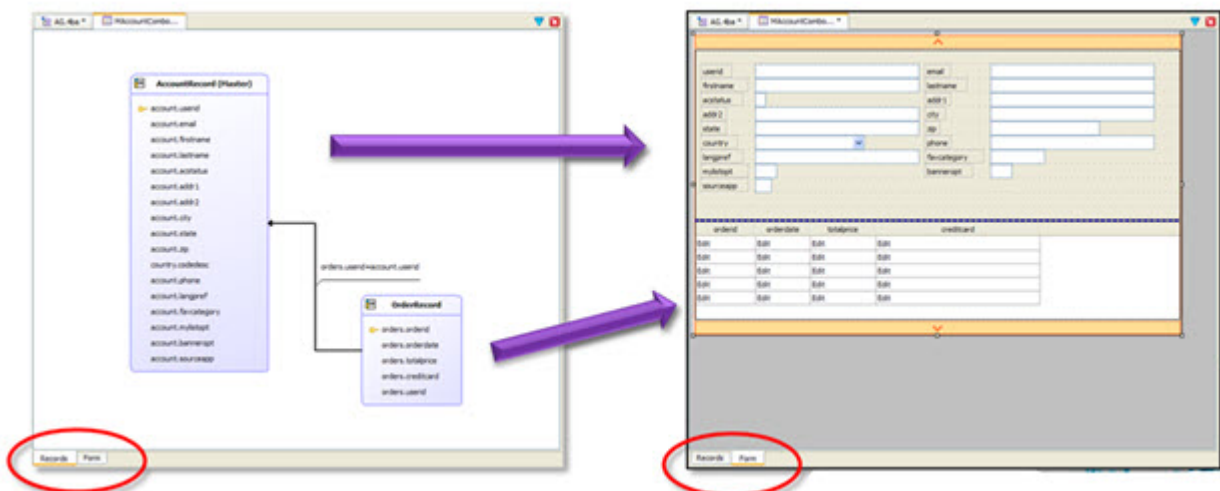


Figure 144: Form Designer diagram

The Form Designer diagram models an application's forms and records. See [Form Designer](#) on page 406.

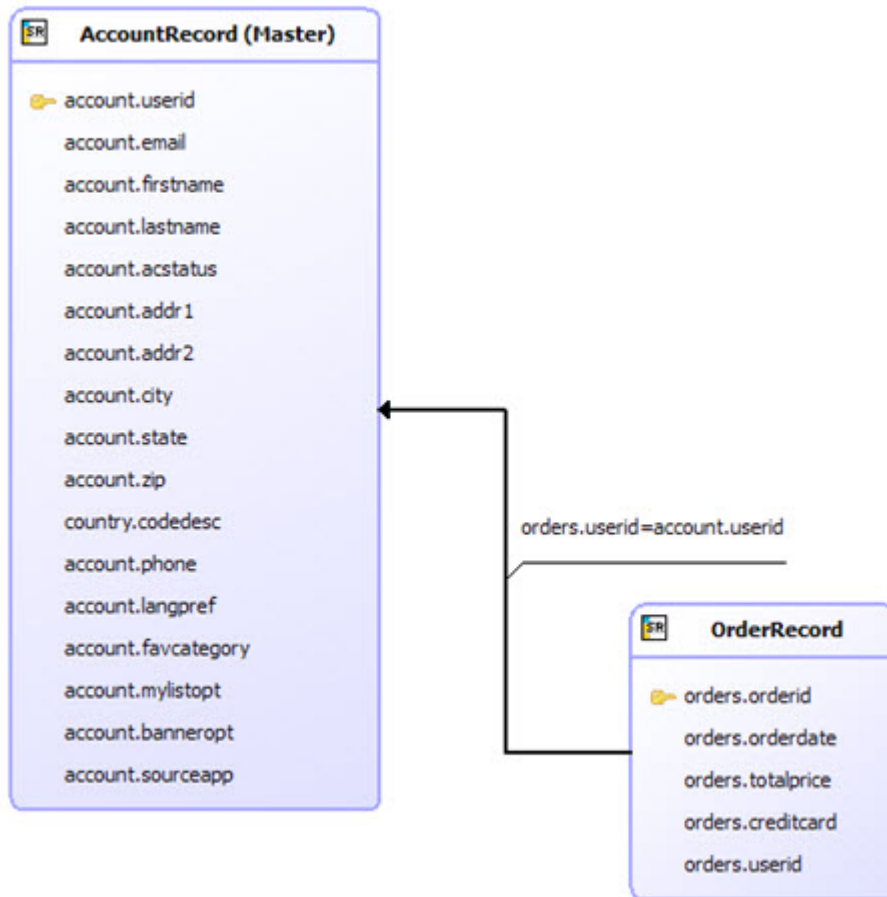


Figure 145: Business Record diagram

Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service.

See [Business records \(data sets\)](#) on page 411.

Mobile apps vs Desktop applications

Mobile apps and desktop applications can be generated using BAM, though mobile apps require a different way of thinking about and organizing an application.

Typical desktop applications often require large, complex master-detail forms, whereas mobile apps require multiple small, simple forms interacting with each other. BAM supports both of these patterns. When modeling the application, you can determine the number of forms and the way in which the forms will relate to each other.

Mobile apps do not support typical reporting and web services, so the options for modeling the app are restricted to program and form entities when working from a mobile project.

The default template features

The default template set is designed to generate organized and functional code for a data-driven business application.

Code architecture

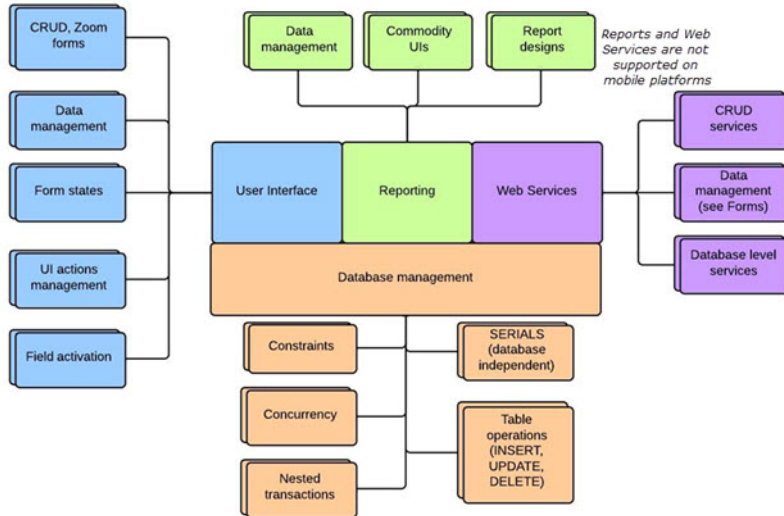


Figure 146: Overview of template features

Table 27: Features

This table lists the general features of the default template set used to generate Genero applications.

Forms	Description
CRUD and Zoom forms	The user interface of a generated program is based on the forms created for the program. There are two types of generated forms - CRUD and Zoom. See Add Forms on page 205.
Form states	See Form behavior in CRUD states on page 241
Functionality	See Enable and disable CRUD logic on page 239
Lookups	See Lookup fields on page 247
Pre-filled comboboxes	The values for a ComboBox list can be automatically retrieved from a database table. Typical usage includes providing a list of state or country values. See Define a dynamically populated ComboBox on page 245
Data management	The generated code manages query by example (QBE), default values, multiple level data sets, all database operations through the database entity (Database meta-schema (4dbx) on page 226), data set key list in memory, and up to date checks before editing. See Modeling the database on page 226.
UI actions management	The generated code manages a Toolbar and menu and the action visibility and enabling, depending on the state of the program. See Default actions on page 264, Default Topmenu and Toolbar on page 267
Field activation	See Field activation on page 244.
Database Management	Description
Constraints	See Database meta-schema (4dbx) on page 226

Database Management	Description
Serials	See Managing SERIALs in a generated application on page 231
Table operations	See Default actions on page 264
Nested transactions	See Understanding what gets generated on page 250
Concurrency	See Managing concurrency on page 231

Reports	Description
Database management	See Implement reports on page 212
Designs	See Add a Report Design Document (4rp) on page 213

Web services	Description
CRUD services	A service can be created either from a CRUD form or from a Zoom form. From a CRUD form the web service can have Create, Read, Update, and/or Delete operations. From a Zoom form, operations are limited to Read. See Create service from a form on page 219.
Database level services	See Modeling the database on page 226

Configuring for BAM

When using the Business Application Modeler, you can specify both a global setup and a template-specific setup.

Factory setup

Factory setup is the standard installation.

Global setup

The global setup overrides the factory setup. It applies to all configurations for the current user for the Genero Studio installation on the current machine or remote machine.

Tools >> Global setup menu option allows you to customize the global setup.

Specific setup

A specific setup is a template-specific setup. It is for the current user, is for a template set used by the Business Application Modeler. Use is option to override the global setup for a specific set of templates. The [GSTSETUPDIR](#) on page 144 environment variable, set in the Genero configuration, defines the template set. If this environment variable is not set, or not specified for the current configuration, the specific setup options are disabled.

The **Tools >> Specific setup** menu option allows you to customize the specific setup.

BAM Projects

Create projects to manage your application development.

- [Managed projects](#) on page 201

- [Mobile projects](#) on page 184

Managed projects

When you create a managed project, nodes for the basic structure of the project are already defined and automatically contain the additional build rules needed for generated programs.

Select **File >> New, Design, Managed Project** to create a managed project. As you add entities to the diagram, you are prompted to save them and add them to a node in the project.

A default managed project contains these nodes.

Group	Used to organize project nodes. Contains nodes of project.
Application	Contains entity files and the generated source code files.
Database	Contains database schemas for the database(s) the application will access.
Library	Contains any additional files used by the application.

Right-click a node in the project to display a menu of options, including adding, renaming, and deleting nodes and files.

Mobile projects

BAM Mobile Projects have predefined nodes for the basic structure of the project. These nodes correspond to the project directories created on disk in the project directory you specified. The project also includes the build rules needed to generate, package, and deploy the application.

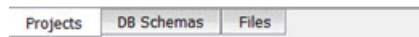
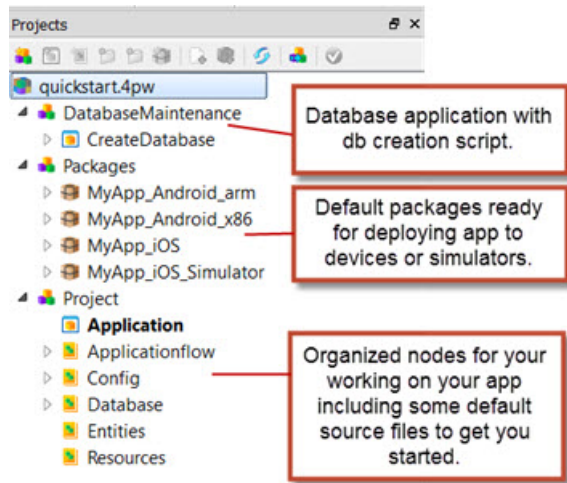


Figure 147: Default Project structure

Table 28: Default Project structure

Node	Files to be saved to this node
Application	4prg (implemented program file from 4ba diagram)
Applicationflow	4ba

Node	Files to be saved to this node
Config	FGLPROFILE
Database	4dbx, db
Entities	4fdm (forms or other entities from 4ba diagram)
Resources	XML files such as 4st for styles

Modeling the application

Model the application by laying out the programs, forms, reports, and relationships between the entities on a BA diagram. Then, implement the programs and forms and reports to specify the details about each entity. Last, build the application to generate the code.

- [The Business Application \(BA\) diagram](#) on page 202
- [Create a BA diagram](#) on page 204
- [Add and implement a program](#) on page 204
- [Add Forms](#) on page 205
- [Add Reports](#) on page 210
- [Add Web services \(Server, Services, Forms with services\)](#) on page 215
- [Add Relations](#) on page 220
- [Add mobile device features \(Photo, Gallery, Phone, Mail, SMS, Contact, Maps, Barcode\)](#) on page 223
- [Import files into the diagram from the project](#) on page 225

The Business Application (BA) diagram

The BA diagram is designed to model the application flow and provide a high-level overview of the application to be generated.

The BA diagram can include various entities to model the application. Relations connect the entities and specify the relationships between them. Each entity has properties to specify all the information needed to generate the code. Until implemented, the entities on the diagram are simply icons. Once implemented, files are created that represent the information and relationships about the entity. These files are used to generate the application code.

Mobile example

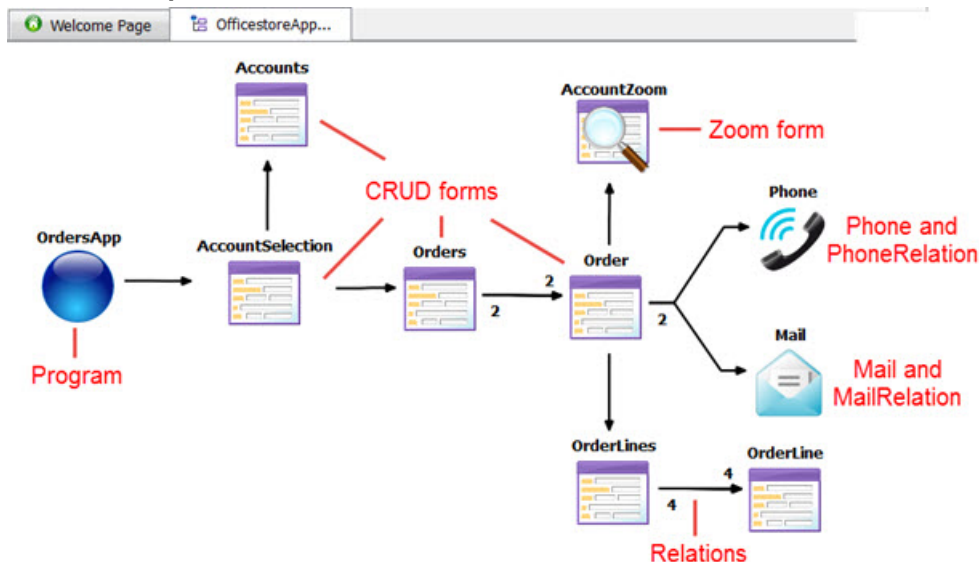


Figure 148: Business Application Diagram (mobile)

Table 29: Entities

Entity	Description
<p>Programs</p>	<p>A Program entity contains the information needed to generate the main logic to drive the application.</p>
<p>CRUD Form entity on page 206</p>	<p>A CRUD Form entity generates the user interface for the program. CRUD forms are used to Create, Read, Update, and Delete data from a database.</p>
<p>Zoom form entity on page 207</p>	<p>A Zoom Form generates a form used to select a value from a list and return the value to the program. It is generally related to a CRUD form field to assist the user with completing the form data entry.</p>
<p>Report entity on page 211</p>	<p>A Report entity generates the logic to retrieve data from the database and run a report based on a defined data definition and report layout.</p>
<p>Webservice Server entity on page 216</p>	<p>Important: This feature is not supported on mobile platforms.</p>
<p>Webservice entity on page 217</p>	<p>A Webservice Server entity contains the main logic to publish services. It listens for incoming requests and executes the relevant service operation.</p>
<p>Webservice entity on page 217</p>	<p>Important: This feature is not supported on mobile platforms.</p>
<p>CRUD form and Webservice, Zoom form and Webservice on page 219</p>	<p>A Webservice entity generates a standalone Webservice.</p>
<p>CRUD form and Webservice, Zoom form and Webservice on page 219</p>	<p>Important: This feature is not supported on mobile platforms.</p>
<p>Relations on page 221</p>	<p>Important: This feature is not supported on mobile platforms.</p>
<p>Phone / PhoneRelation</p>	<p>A Relation entity is used to define a relationship between entities on the BA diagram. Multiple relations can be set to the same form providing easier maintenance, but also flexibility on what CRUD operations are available to the form when it is opened by different actions.</p>
<p>Phone / PhoneRelation</p>	<p>A Phone and PhoneRelation entity is used to define that the form will generate code to launch the default phone app on the device and initiate dialing of the specified phone number.</p>
<p>Mail / MailRelation</p>	<p>A Mail and MailRelation entity is used to define that the form will generate code to invoke the user's default mail application for a new mail to send.</p>
<p>Mail / MailRelation</p>	<p>A Mail and MailRelation entity is used to define that the form will generate code to invoke the user's default mail application for a new mail to send.</p>
<p>Gallery / GalleryRelation</p>	<p>A Gallery and GalleryRelation entity is used to define that the form will generate code to let the</p>
<p>Gallery / GalleryRelation</p>	<p>A Gallery and GalleryRelation entity is used to define that the form will generate code to let the</p>

Entity	Description
Photo / PhotoRelation	user select a picture from the mobile device's photo gallery and return a picture identifier. A Photo and PhotoRelation entity is used to define that the form will generate code to let the user take a picture with the mobile device and return the corresponding picture identifier.
SMS / SMSRelation	An SMS and SMSRelation entity is used to define that the form will generate code to send an SMS text to one or more phone numbers.
Contact / ContactRelation	A Contact and ContactRelation entity is used to define that the form will generate code to let the user choose a contact from the mobile device contact list and return the vCard.
Map / MapRelation	A Map and MapRelation entity is used to define that the form will generate code to invoke a maps app with the current Global Positioning System (GPS) location of the mobile device.
Barcode / BarcodeRelation	A Barcode and BarcodeRelation entity is used to define that the form will generate code to use a barcode scanner from the mobile device.

Create a BA diagram

The BA diagram is designed to model the application flow and provide a high-level overview of the application to be generated.

1. Select **File >> New, Design, Application Modeling, Business Application Diagram**.
2. Right-click in the background of the diagram to display a context menu of options. Add entities such as Programs, CRUD forms, Zoom forms, Reports, and Web Services. Repeat this step until you have added all entities that model your application.
3. Define the relationships between the entities with a relation entity. Right-click on an entity, such as a Program, and select **New Relation**. Click and drag the Relation arrow to the entity to which you want to set the relationship. Repeat this step until all relationships have been defined.
4. Set properties on the entities.

Add and implement a program

Implementing a program entity on the BA diagram creates the `4prg` file which is used to generate the program code.

1. Open the BA diagram to which you want to add a program.
2. Right-click on the diagram and select **New, Program** to add a new Program entity.
3. From the BA diagram, select a program entity.
4. Right-click and select **Implement Program**. Save and name the `4prg` file and optionally select to insert it into a location in the project.
The `4prg` file is in the project along with placeholders in the Intermediate Files folder for the xml and 4gl source files that will be generated for the program.

Program entity

A **Program** entity contains the information needed to generate the main logic to drive the application.

A Program entity is represented as a `4prg` file.

When a program entity is implemented, a 4gl source file is generated which contains the `MAIN` function for the application. This 4gl file is also used to combine the other generated 4gl files into a Genero application.

Any changes to the diagram are included in the subsequent re-generating of the program code.

The generated code can be customized.

Table 30: Entity Properties

Property	Description
Name	Name of entity.
Type	Type of entity.
File Name	Full path to file.
Description	Description of the program entity.

Right-click a Program entity in the diagram to display a context menu of options.

Table 31: Context Menu

Menu Option	Description
Implement Program	Implementing a program entity on the BA diagram creates the 4prg file which is used to generate the program code. Until you create the Program 4prg file, the Program entity on the diagram is simply an icon.
New Relation	Add a relationship to another entity on the diagram.
Execute Program / Build Program	Builds and/or executes the source code for the program based on the related entities in the diagram.
Rename	Changes the name property of the entity.
Convert to	Converts entity from one type to another.
Locate in Project	Locates and highlights the selected item in the project.
Hide / Show all Items	Hides the entity from view. Show again by selecting the Show all Items option.
Filter Items	The Filter View dialog allows you to hide and show items on a diagram.
Delete	Deletes the entity.
Select All	Select all entities or all entities of the same type on the diagram.

Add Forms

The user interface of a generated program is based on the forms created for the program. There are two types of generated forms - CRUD and Zoom.

CRUD

CRUD forms are used to Create, Read, Update, and Delete data from a database. CRUD forms are typical data input and browsing forms. One CRUD form can call another CRUD form or a Zoom form.

Zoom

A **Zoom form** generates a form used to select a value from a list and return the value to the program. It is generally related to a CRUD form field to assist the user with completing the form data entry. For example, a Zoom form could be created that displays a list of country codes from

the **country** database table. When the user triggers the zoom action to occur, either by selecting the zoom icon on the Toolbar or from the zoom field, a list of country codes displays in a popup window. When a country code is chosen from the list, the zoom form closes and the value is populated in the corresponding field on the main form.

CRUD Form entity

A **CRUD Form** entity generates the user interface for the program. CRUD forms are used to Create, Read, Update, and Delete data from a database.

A CRUD form entity is represented as a `4fdm` file which can be opened and edited in Form Designer.

When a CRUD form entity is implemented from the diagram, the `4fdm` file is created as well as `4gl` source files containing the program logic to access and manipulate the database tables contained in the form.

Any changes to the CRUD form entity properties in the BA diagram or in the `4fdm` file are included in the subsequent re-generating of the program code.

The generated code can be customized.

Properties can be set specify how the form should behave during the [various states \(DISPLAY, MODIFY, ADD, SEARCH\)](#).

If your form has a valid relation on the BA diagram to a Report entity, additional actions are generated for the Toolbar and Topmenu to launch the report. The Report Option properties are used to define which report actions should be generated.

Important: This feature is not supported on mobile platforms.

Table 32: Report Option Properties

Property	Description
quickPrint	Defines if print action is available.
quickPreview	Defines if preview action is available.
quickPDF	Defines if export to PDF action is available.
quickHTML	Defines if export to HTML action is available.
quickXLS	Defines if export to XLS action is available.
quickRTF	Defines if export to RTF action is available.
canExport	Defines if the PDF, HTML, XLS, and/or RTF print actions are available. If canExport is checked and quickPDF, quickHTML, quickXLS, or quickRTF are also checked, the action will be available.

Right-click the Form entity in the diagram to display a context menu of options.

Table 33: Context Menu Options

Menu Option	Description
Implement CRUD Form	Creates a new blank managed form definition file (<code>4fdm</code>) to design from scratch.

Menu Option	Description
Implement CRUD Form from Database	Provides a wizard to create a managed form definition file (4fdm), allowing you to pick the columns and general display of your form. This option is generally preferred. The form can be modified after it is created.
Open Form	Opens form in Form Designer.
Convert to	Converts entity from one type to another.
Locate in Project	Locates and highlights the selected item in the project.

Add CRUD forms

A generated application can include multiple forms.

This task assumes that you have created a [BA diagram](#) with a Program entity.

1. Right-click on the open BA diagram and select **New CRUD Form**.
2. Right-click the new form and select **Implement CRUD Form from Database** from the context menu.
3. Select the database table and fields that you want to have on the new form (the products table, in our example). Finish and save the form definition file (4fdm) to your project.
4. To call one form from another, create a second CRUD form by repeating steps 2 and 3.
5. Right-click the first form icon and select **New Relation**. Click the Form icon again and drag the relation arrow to the second form.
6. Select the relation arrow in the BA diagram and set the **action** property in the Properties view to the action name that you wish to trigger (**products** in our example).
7. Save the forms and modified BA diagram.
8. Build and run the program. A button is displayed on the calling form to trigger the action to call the secondary form. This button is added at runtime and can be removed by adding the action to the Toolbar and/or Topmenu. To do so, modify the default Toolbar (4tb) and/or Topmenu (4tm) files. Modify the action's display attributes by modifying the action defaults file (4ad).

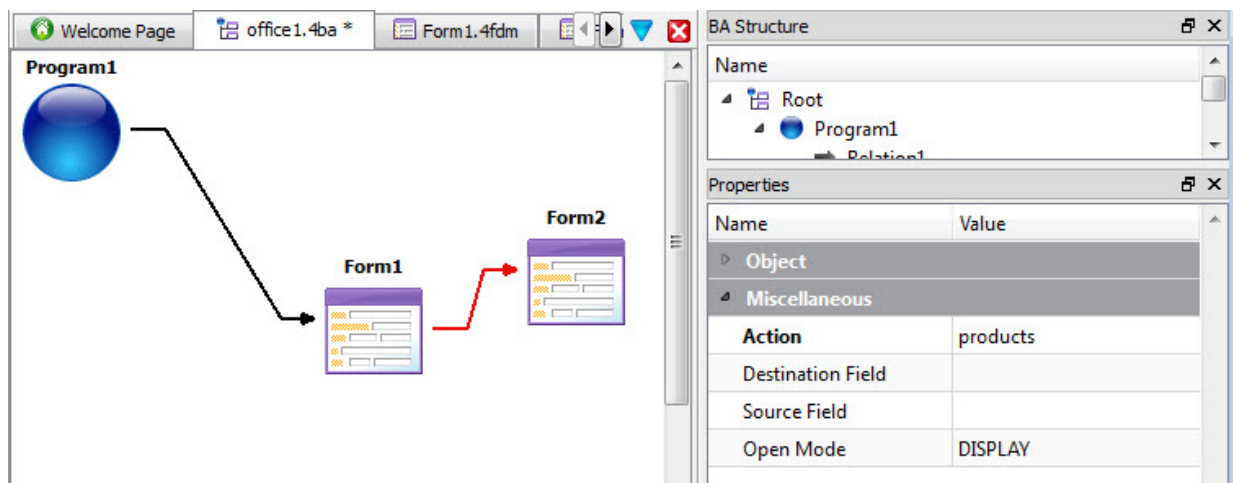


Figure 149: BA diagram with two forms

Zoom form entity

A **Zoom Form** generates a form used to select a value from a list and return the value to the program. It is generally related to a CRUD form field to assist the user with completing the form data entry.

A Zoom form entity is represented as a 4fdz file which can be opened and edited in Form Designer.

When a Zoom form entity is implemented from the diagram, the 4fdz file is created as well as 4g1 source files that contain the logic to allow a user to pick a value from a list that is displayed in a form in a popup window.

Any changes to the Zoom form entity properties in the BA diagram or in the 4fdz file are included in the subsequent re-generating of the program code.

The generated code can be customized.

Properties can be set specify how the form should behave during the [various states \(DISPLAY, SEARCH\)](#).

Right-click the Zoom Form entity in the diagram to display a context menu of options.

Table 34: Context Menu Options

Menu Option	Description
Open Form	Opens form in Form Designer.
Implement Zoom	Creates a new blank zoom form definition file (4fdz) to design from scratch.
Implement Zoom from Database	Provides a wizard to create a zoom form definition file (4fdz), allowing you to pick the columns and general display of your form. This option is generally preferred. The form can be modified after it is created.
Convert to	Converts entity from one type to another.
Locate in Project	Locates and highlights the selected item in the project.
Filter Items...	The Filter View dialog allows you to hide and show items on a diagram.

Add Zoom forms

Zoom forms contain logic that allow the user to pick a value from a list that is displayed in a form in a popup window.

This task assumes that you have created a [BA diagram](#) with a Program entity and at least one [CRUD form entity](#).

1. Right-click on the open BA diagram and select **New Zoom Form**.
2. Right-click the new form and select **Implement Zoom from Database** from the context menu.
3. Select the database table (in our example, the **country** table) that contains the values to be displayed in the zoom form. Select the column that corresponds to the field on the main form. In our example, this is the country code. (You can select the country name also, to display it in the zoom form if you wish; the code field will be identified as the unique key later in this process.)

Country code	Country
Edit	Edit
Edit	Edit
Edit	Edit
Edit	Edit
Edit	Edit

Figure 150: Example zoom form definition

4. Switch from the form design to the **Records** tab and select the master table record. Confirm that the active property is checked.
5. Select the record and confirm that the `unique key` property is set. If not, set the unique key on one of the fields in the record.

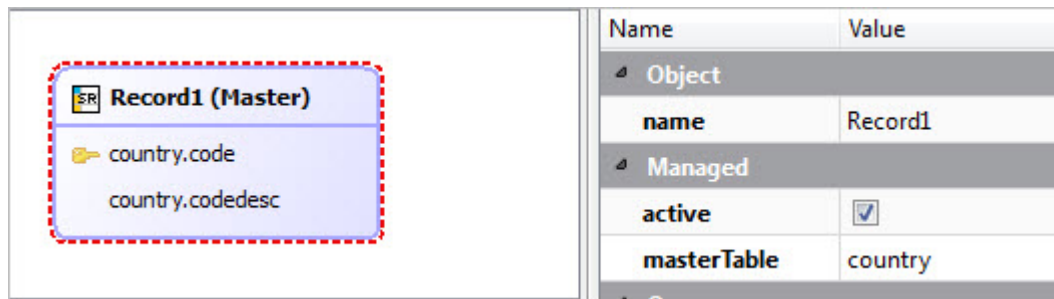


Figure 151: Records tab.

6. Save the Zoom form (4fdz) and add it to the project structure under the Application node.
7. The CRUD form has to be modified to trigger the zoom form when needed. In this example, the country.code field is changed to a ButtonEdit, which can trigger an action when the user clicks it. When the user selects the country code from the zoom form, it is automatically inserted into the country.code field on the main form. From the BA diagram, right-click on the main form and choose **Open Form**.
8. Select the field containing the foreign key (in this example, the **account.country** field.) Right-click and select **Convert Widget** to change the widget to a ButtonEdit.

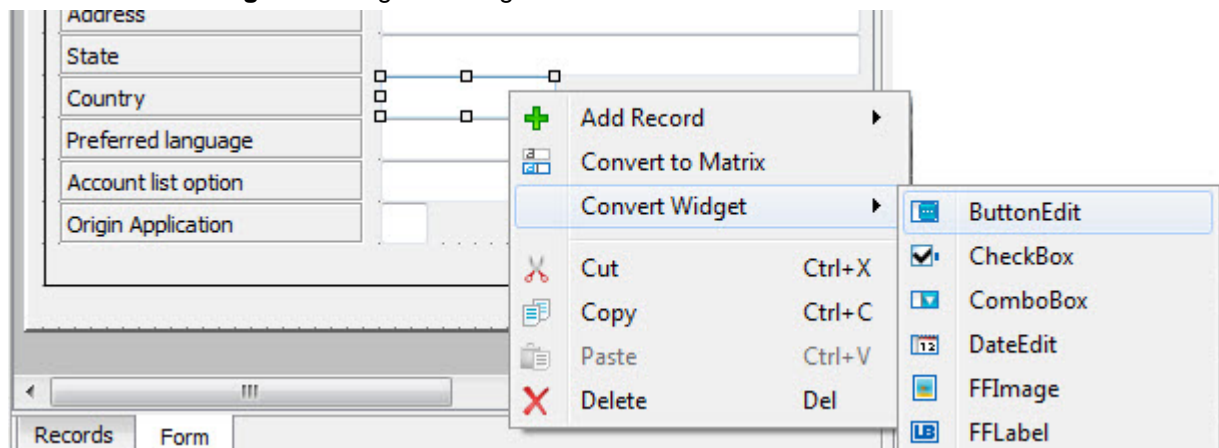


Figure 152: Convert Widget

9. Scroll to the bottom of the properties list, and set the value of the **action** property for the ButtonEdit field to a unique action name (such as **zoom**).
10. Save the form.
11. Create a relation between the CRUD form and the Zoom in the BA diagram. On the open BA diagram, Right-click the CRUD form icon and select **New Relation**. Click the CRUD form icon again and drag the relation arrow to the second form.
12. Select the relation arrow to display the relation properties, and enter the action name in the **action** property that is the same as the one assigned to your ButtonEdit field (**zoom1**, in this example.) If there is more than one zoom form in an application, the action name must be unique.

Option	Description
Source Field	Set to the value that will be inserted into on the CRUD form (account.country, in this example).
Destination Field	Set to the value the user selects from the table in the zoom form (country.code, in this example).
Open Mode	Set to DISPLAY (default) or SEARCH.

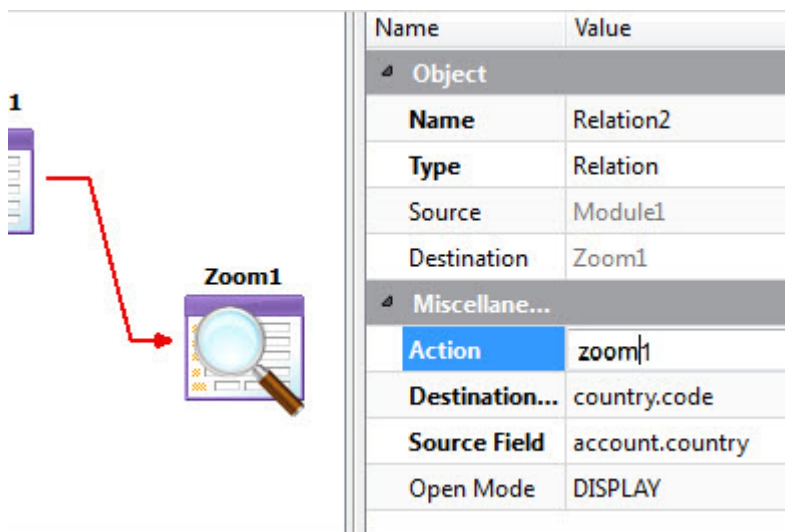


Figure 153: Business Application diagram

13. Save all diagrams and forms.

Implement a form

Implementing a form entity on the BA diagram creates the `4fdm` or `4fdz` file which is used to generate the form code.

1. From the BA diagram, select a CRUD or Zoom form entity.
2. Right-click and select an option to implement the form.

Option

Description

Implement CRUD Form

Creates a new blank managed form definition file (`4fdm`) to design from scratch.

Implement CRUD Form from Database

Provides a wizard to create a managed form definition file (`4fdm`), allowing you to pick the columns and general display of your form. See [Data Control wizard](#) on page 449.

Implement Zoom Form

Creates a new blank zoom form definition file (`4fdz`) to design from scratch.

Implement Zoom Form from Database

Provides a wizard to create a zoom form definition file (`4fdz`), allowing you to pick the columns and general display of your form. See [Data Control wizard](#) on page 449.

3. Save the form to your project.

Placeholder `4gl` and `xml` files are created in the Intermediate Files folder in the project. When the form or project is compiled, these files are populated with the logic for your form.

Add Reports

You can add a Genero Report Writer report to your generated application.

1. Open the BA diagram to which you want to add a report.
2. Right-click on the diagram and select **New Report Data** to add a new Report Data entity.
3. Right-click on the Form entity from which you want to access the report and select **New Relation**. Click on the Form entity again and drag the relation link from the Form entity to the Report Data entity.
4. Select the relation link between the Form and Report Data entities. In the Properties panel, change the **Type** property to **ReportRelation**.

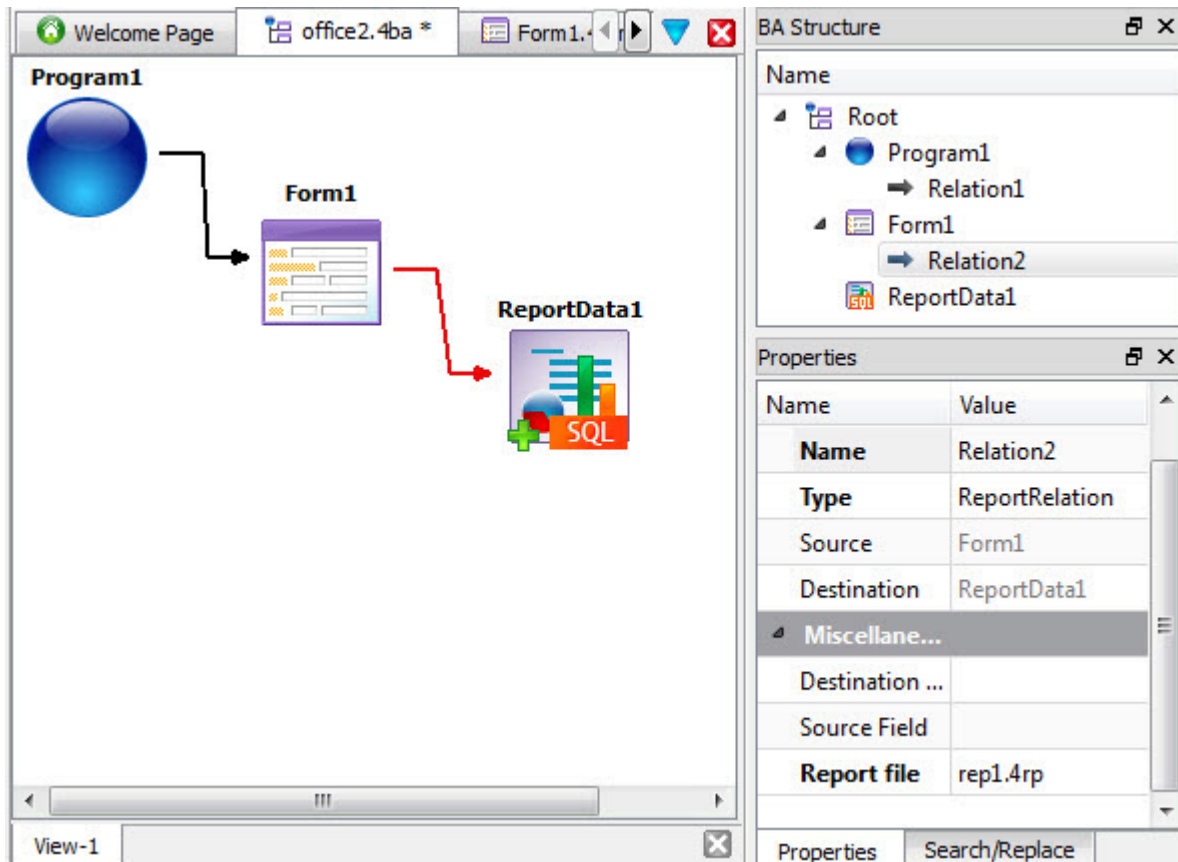


Figure 154: Adding a Report entity to the BA Diagram

Report entity

A **Report** entity generates the logic to retrieve data from the database and run a report based on a defined data definition and report layout.

A Report entity is represented as a 4rd file which can be opened and edited.

When a Report entity is implemented from the diagram, the 4rd file is created as well as 4gl source files containing the report's driver and routing instructions.

Any changes to the Report entity properties in the BA diagram or in the 4rd file are included in the subsequent re-generating of the program code.

The generated code can be customized.

Right-click the Report Data entity in the diagram to display a context menu of options.

Table 35: Context Menu Options

Menu Option	Description
Implement Report Data	Creates a new blank Report Data entity (4rd) to build from scratch.
Implement Report Data from Database	Provides a New Record wizard to select tables and columns to include in the Report Data entity business record (4rd). This option is generally preferred. The business record can be modified after it is created.
Open Business Record	Opens report business record.
Convert to	Converts entity from one type to another.
Locate in Project	Locates and highlights the selected item in the project.

Implement reports

The report record contains the data definition, structure, and table relationships required to generate a `rdd` (Report Data definition file). The `rdd` file is used in conjunction with a Genero report definition (`4rp`) file to automatically generate the reports.

1. Right-click on the Report Data entity and select **Implement Report Data from Database**. Select the desired tables and columns.
2. Select the business record to display its properties. Make sure that the active property is checked and that the `masterTable` property contains the database table name.
3. If more than one table was added to the record, select the Query property and specify the joins between the tables in the Query Editor.
4. Confirm that the unique key property is set on the field in the record that represents the primary (unique) key in the database table.
5. Save the Report Data file (`4rd`) to your project.
6. Build the application. The needed data definition file (`rdd`) is generated for you.
7. Run the application. Notice that additional actions have been added to the Toolbar and Topmenu for reports.
8. Select Preview from the Toolbar to view the report. The report will run using the default layout. You can add a Report Design Document (`4rp`) to customize the look and feel of the report.

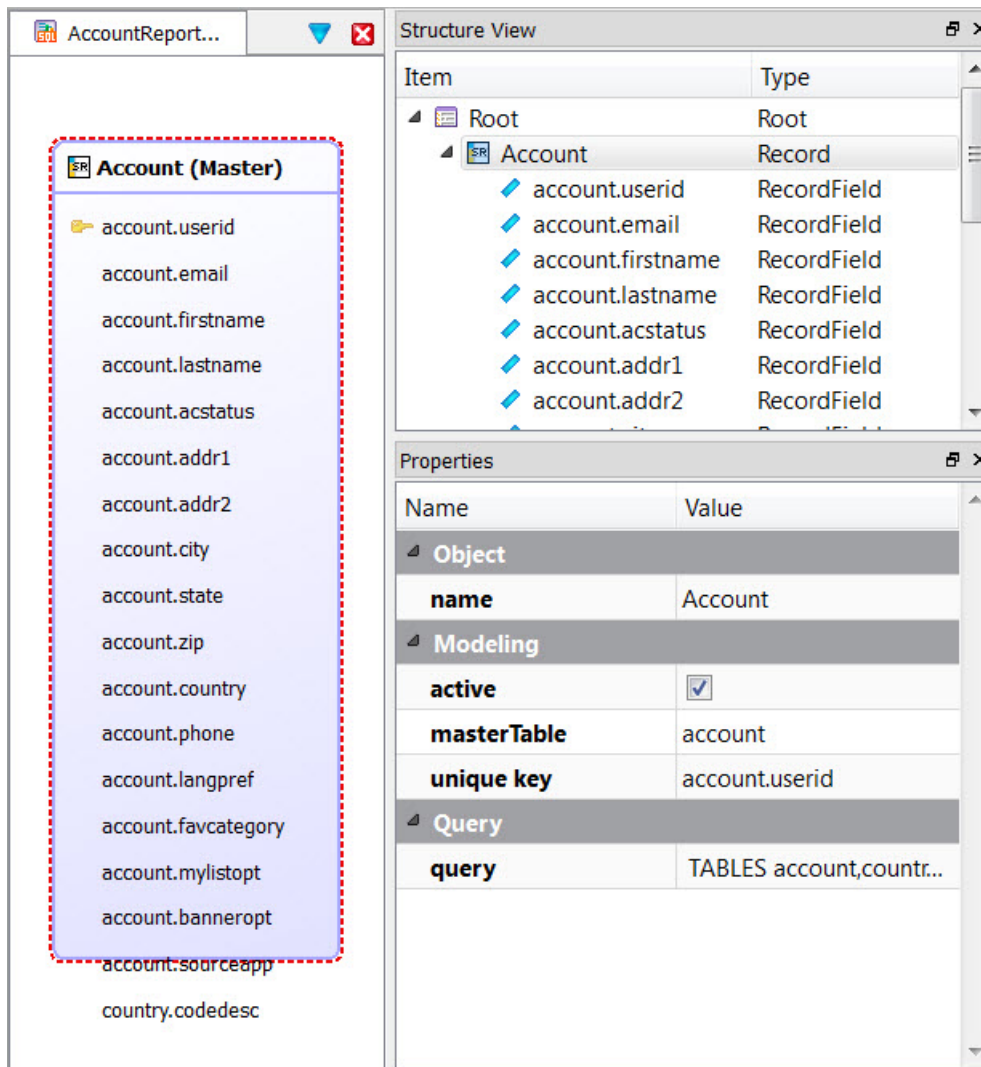
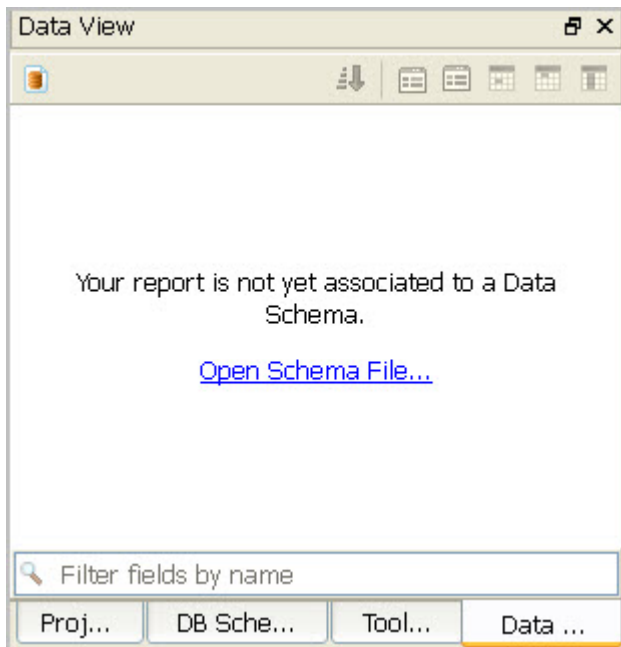


Figure 155: Populated Record

Add a Report Design Document (4rp)

You can customize the look and feel of your report with a Report Design Document (4rp).

1. From the Genero Studio main menu, select **File>>New, Reports, Report Designs** and select the type of report you wish to create.
2. Select the **Data View** tab from the Project View.
3. Select **Open Schema File...** to associate the generated report data definition file (rdd) with this report design. You can find the name of the generated rdd file in your project's **Intermediary Files** listing.



4. [Design your report](#) by dragging and dropping fields from the Data View tab to your report design.
5. Save and name the Report Design Document (4rp) to your project, such as `listreport.4rp`.
6. Return to the BA diagram and select the relation link between the Form and Report Data objects.
7. Supply the name of your Report Design Document, such as `listreport.4rp`, for the **Report File** property.
8. Build and run the application and select Preview from the Toolbar to view the report with the new design.

Report print settings

The **Report Print Settings** dialog appears when the reportsetup action is triggered from a generated application.

By default, the reportsetup action is bound to the **Print...** button in the generated user interface Toolbar.

Options on the dialog are controlled by properties set on the [ReportRelations](#), the relationships between the current Form object and one or more Report Data entities on the Business Application diagram (4ba).

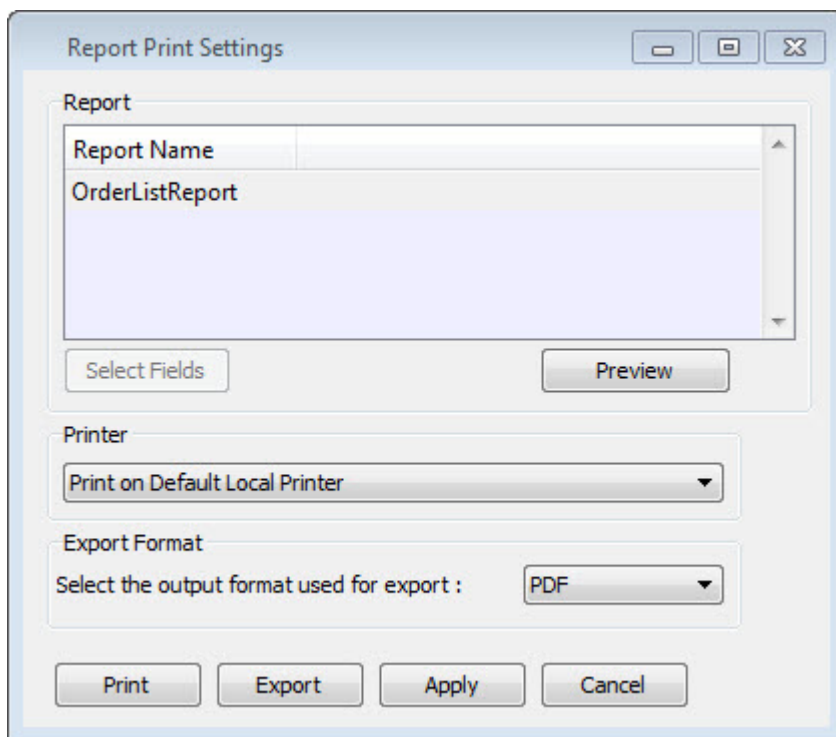


Figure 156: Report Print Settings

Table 36: Report Print Settings Options

Option	Description
Report List	The Report List shows all Report Label values for ReportRelations to the current Form.
Select Fields	Select Fields is enabled if there is not a Report Designer (4rp) file specified in the Report File property for the ReportRelation to the current Form. Select Fields is then used to select the fields to print on the report.
Preview	Displays the selected report in the Genero Report Viewer.
Printer	Select printer to print report. Options are Default Local Printer or Choose a Local Printer.
Export Format	Export selected report to PDF, HTML, XLS, or RTF.

Add Web services (Server, Services, Forms with services)

Web services are a standard way of communicating between applications over an intranet or Internet. They define how to communicate between two entities: a server that exposes services and a client that consumes services. Web services may provide programmable access to the functions of a form or a standalone service without a form.

- [Webservice Server entity](#) on page 216
- [Create a Webservice server](#) on page 216
- [Webservice entity](#) on page 217
- [Create standalone service](#) on page 219
- [Create service from a form](#) on page 219

- [CRUD form and Webservice, Zoom form and Webservice](#) on page 219
- [JSON Web services](#) on page 220
- [Public fields](#) on page 220

Webservice Server entity

The **Webservice Server** entity contains the main logic to publish services. It listens for incoming requests and executes the relevant service operation.

The XML representation of a Webservice Server entity is a `4wsprg` file.

When a Webservice Server entity is implemented from the diagram, a `4gl` source file is generated which contains the `MAIN` function for the application. This `4gl` file is also used to combine the other generated `4gl` files into a Genero application.

Any changes to the BA diagram are included in the subsequent re-generating of the program code.

The generated code can be customized.

Table 37: Entity Properties

Property	Description
Name	Name of entity.
Type	Type of entity.
File Name	Full path to file.
Namespace	Common namespace used for the published services.

Right-click a Webservice Server entity in the diagram to display a context menu of options.

Table 38: Context Menu Options

Menu Option	Description
Implement Webservice Server	Creates the Webservice Server file (<code>4wsprg</code>). The <code>4wsprg</code> generates the logic for the <code>MAIN</code> function. Until you create the Webservice Server file, the Webservice Server entity on the diagram is simply an icon.
Execute Program / Build Program	Builds and/or executes the source code for the Webservice Server based on the related entities in the diagram.
Convert to	Converts entity from one type to another.
Locate in Project	Locates and highlights the selected item in the project.

Create a Webservice server

A Web service server is in charge of publishing services. It listens for incoming requests and executes the relevant service operation.

1. Open the BA Diagram, right-click on the background of the diagram and select **New>>Webservice Server**. Optionally, specify a common namespace used for published services.
2. Draw a relation from the Webservice Server to each Web Service or Form with Web Service entity that are to be connected to the Webservice Server.
Service publication is done by creating the relations between the server entity and the form with service or service standalone entities.
3. Select the Webservice Server and select **Implement Webservice Server**. Save and name the server in your project.

Webservice entity

Web services are a standard way of communicating between applications over an intranet or Internet. They define how to communicate between two entities: a server that exposes services and a client that consumes services.

A Webservice entity generates a web service with its CRUD operations, but with no accessible form. CRUD operations are used to Create, Read, Update, and Delete data from a database. It includes one global *read* operation that reads all records in once and one global *create* operation that creates all records at once.

A standalone Webservice entity is represented as a `4ws` file which can be opened and edited in the Form Designer Records tab. When a **Webservice** entity is implemented from the diagram, the `4ws` file is created as well as `4gl` source files containing the program logic to create and set up a web service and its CRUD operations. Any changes to the Webservice entity properties in the BA diagram or in the `4ws` file are included in the subsequent re-generating of the program code. The generated code can be customized.

See the topic *Introduction to Web Services* in the *Genero Business Development Language User Guide* for more information on Web Services concepts.

Properties of the Webservice entity

Select the Webservice entity in the Business Application diagram to view and set the entity properties.

Table 39: Webservice entity Properties, Object Category

Group	Property	Description
Object	Name	Name of entity.
	Type	Type of entity: <ul style="list-style-type: none"> WebService (for a SOAP Webservice). WebServiceJSON (for a JSON Webservice).
	File Name	Full path to file.
	Description	Description of web service.
Web Service	Service Name	Unique publishable service name.
	Comment	Service comment that will appear in the <code>wsdl</code> file.

Contextual menu for the Webservice entity

Right-click a Webservice entity in the diagram to display a context menu of options.

Table 40: Webservice entity context menu options (partial listing)

Menu Option	Description
Implement Web Service	Creates a new blank managed web service definition file (<code>4ws</code>) to design from scratch. Note: Once created, this menu option disappears; use Open for any future modifications.
Implement Web Service from Database	Provides a wizard to create a managed web service definition file (<code>4ws</code>), allowing you to pick the columns to use in the CRUD operations. This option is generally preferred. The web service can be modified after it is created. Note: Once created, this menu option disappears; use Open for any future modifications.

Menu Option	Description
Open	Open the Business Record for the selected Webservice entity.
Convert to	Converts entity from one type to another.
Locate in Project	Locates and highlights the selected item in the project.

Webservice entity Business Record

Once the Web service definition file (. *4ws*) exists, right-click the Webservice entity and select **Open** to view the Business Record diagram.

In the Structure view, select the **Root** node. All Web services share the same set of properties in the Root node.

Object group name, databaseName

In the Structure view, select a **Record** node. All Web services share the same Record node properties for the following groups:

Object group name

Modeling group active, masterTable, unique key

Query group query

Functionality group canDisplay, canAdd, canModify, canDelete, canSearch

For the same Record node, however, only SOAP Web services contain the properties that provide for XML and XSD Schema Serialization attributes:

Web Service group XMLAll, XMLSequence, XMLList, XMLElementNamespace, XMLAttributeNamespace, XSTypename, XSTypenamespace

See the section on XML serialization in the *Genero Business Development Language User Guide* for more information.

In the Structure view, select a **RecordField** node. All Web services share the same RecordField node properties for the following groups:

Object group name

Modeling group lookup

Field group fieldType, sqlTabname, colName, fieldIdRef, dataType

Web Service group public

Miscellaneous group defaultValue

For the same RecordField node, however, only SOAP Web services contain the properties that provide for XML and XSD Schema Serialization attributes:

Web Service group XMLOptional, XMLElement, XMLAttribute, XMLName, XSDType, XSDLength, XSDMinLength, XSDMaxLength, XSDEnumeration, XSDWhiteSpace, XSDPattern, XSDMinInclusive, XSDMaxInclusive, XSDMinExclusive, XSDMaxExclusive, XSDTotalDigits, XSDFractionDigits

See the section on XML serialization in the *Genero Business Development Language User Guide* for more information.

Create standalone service

The standalone service provides CRUD operations on specified business records (data sets).

1. Create a new Web service.
 - From an open BA diagram, right-click on the background and select **New Webservice**.
 - Alternatively, select **File >> New >> Design, Web Service** or **Web Service from Database**.
2. Right-click on the service to implement it. Create the business records that you want to handle in the services.
3. Save the file in your project. (This project can now be imported into a BA diagram using the **Import Project Files to BA Diagram** menu option.)
4. Functionality properties can be set on each record to specify whether the services operation of add, update, delete and/or search should be generated.

Create service from a form

A service can be created either from a CRUD form or from a Zoom form. From a CRUD form the web service can have Create, Read, Update, and/or Delete operations. From a Zoom form, operations are limited to Read.

1. Open the BA Diagram and select a form.
2. In the properties view, enter the service name into the service name property. Service CRUD operations apply on the records (data sets) that are already defined in the form. A new data set is not necessary, but the data set can be customized.
3. Compile the form. The service is created during form compilation.

CRUD form and Webservice, Zoom form and Webservice

Forms with web services allow programmable access to a form's functionality through a published service.

A **CRUD form and Webservice** entity generates a CRUD form and a standalone Webservice and is represented as a `4fdmws` file.

A **Zoom form and Webservice** entity generates a Zoom form and a standalone Webservice and is represented as a `4fdzws` file.

These files can be opened and edited in Form Designer. They have all the properties of a [CRUD Form entity](#) on page 206 or [Zoom form entity](#) on page 207 plus `Service Name` and `Comment`.

Note: Only Read operation is generated for a Zoom form and Webservice entity.

Table 41: Unique Properties

Property	Description
Service name	Unique publish-able service name.
Comment	Service comment that will appear in the <code>wsdl</code> file.

Right-click the entity in the diagram to display a context menu of options.

Table 42: Context Menu Options

Menu Option	Description
Implement CRUD/Zoom Web Service	Creates a new blank managed web service definition file to design from scratch.

Menu Option	Description
Implement CRUD/ Zoom Web Service from Database	Provides a wizard to create a managed web service definition file, allowing you to pick the columns to use in the CRUD operations. This option is generally preferred. The web service can be modified after it is created.
Convert to	Converts entity from one type to another.
Locate in Project	Locates and highlights the selected item in the project.

JSON Web services

You can generate a JSON Web service.

JSON Web Services are modeled in a similar way as SOAP Web Services with:

- JSON Server
- JSON Services
- JSON Form + Service
- JSON Zoom Form + Service

The exchange protocol is HTTP/REST with JSON encoded messages.

For JSON Services, both the server implementation and a client (consumer) API are generated from the model.

In order to write a client application, just link the generated client stub with the service consumer application and the files `libdbappWSCore.42m` and `libdbappWSClient.42m` of the `dbapp` library.

Public fields

A public field is a field that appears in the service API.

Only public fields are treated in the service CRUD operations. By default all fields in a data set are public.

Add Relations

Define a relationship from one entity to another with a relation entity.

1. Right-click on an entity, such as a Program, and select **New Relation**.
2. Click and drag the Relation arrow to the entity to which you want to set the relationship, such as a Form. If you try to set an invalid relation, for example from a Form entity to a Program, you will see the constraint icon

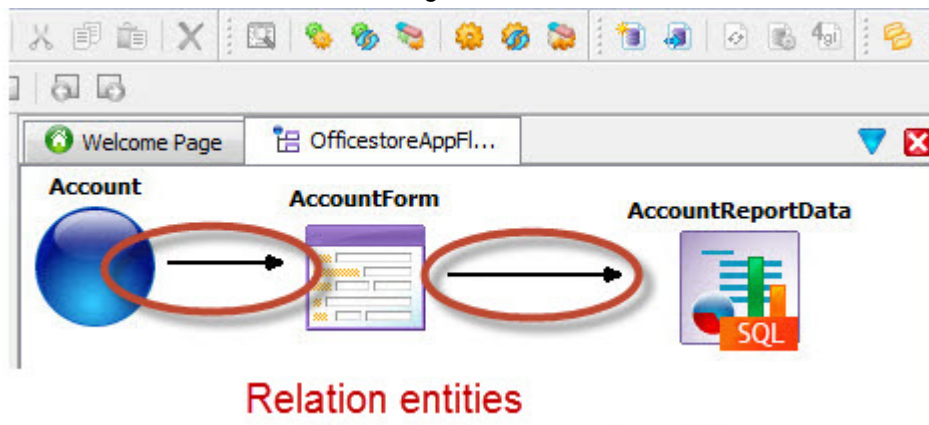


indicating that this relation is not valid.

3. Repeat this step until all relationships have been defined.

Relations

A **Relation** entity is used to define a relationship between entities on the BA diagram. Relations are represented with arrows on the BA diagram.



There are specific types of relations:

Relation

Relation between Program and Form entities and between Form entities. A form can have multiple relations to another form, thus allowing the same form to behave differently depending on how it is used in the application. See [UI Settings](#).

ReportRelation

Relation between Form and Report entities.

WebServiceRelation

Relation between Web Service Server and Web Service entities.

PhotoRelation

Relation to Photo entity in mobile app diagram.

GalleryRelation

Relation to Gallery entity in mobile app diagram.

PhoneRelation

Relation to Phone entity in mobile app diagram.

SMSRelation

Relation to SMS entity in mobile app diagram.

ContactRelation

Relation to Contact entity in mobile app diagram.

MapsRelation

Relation to Maps entity in mobile app diagram.

BarcodeRelation

Relation to Barcode entity in mobile app diagram.

Note: See [Mobile device function properties](#) on page 223 for properties specific to mobile app entity relations.

Table 43: Entity Properties

Property	Description
Name	Name of relation.
Type	Type of relation.
Source	Relation source entity name.
Destination	Relation destination entity name.
Action	Action name to be used in generated code to trigger relationship. For example, "zoom1" might be an action name between a Form and a Zoom Form entity.

Property	Description
Source Record	If an action is desired only for a specific subdialog, this action is set to a specific record in the form. When empty, the action defined on the relation is global to the <code>DIALOG</code> except if <code>Row Bound</code> property is checked.
Row Bound	Renders a contextual action to the current row. See Rowbound actions on page 237. If a form has more than one record, <code>Row Bound</code> must be set to one of the records.
Source Field (Position)	Specifies the fields used to locate a specific row when opening a Form . They usually correspond to database Foreign Keys. They must match in number and type the Business Record unique key of the target Form. See Control the row position in form on page 244.
Source Field (Filter)	Specifies the fields whose values are used for filtering when opening a Form. See Opening a form with a subset of data on page 244.
Destination Field (Filter)	Specifies the fields to filter on when opening a Form. See Opening a form with a subset of data on page 244.
Open Mode	Open Mode is the initial state of the form when opened. The rendered form's default Toolbar allows the user to switch modes.
Default Mode	Default Mode is the mode in which you return after leaving another mode.
Data Refresh	If the source form is in <code>DISPLAY</code> mode and opens a destination form, when the destination form closes, the data displayed in the source form refreshes. See Data refresh on page 243.
Functionality	Change the form's behavior in this relation. See Form behavior in CRUD states on page 241.
Target Behavior	Change the form's behavior in this relation. See Form behavior in CRUD states on page 241.
Report File	ReportRelation type relations only. Names the Report Design Document (<code>4rp</code>) to be used for the report. If blank, user can use Select Fields option on Print Report Settings dialog to choose fields for report output.
Report Label	Name of report shown in print list.

Right-click the Program entity in the diagram to display a context menu of options.

Table 44: Context Menu Options

Menu Option	Description
Filter Items ...	The Filter View dialog allows you to hide and show items on a diagram.
Delete	Deletes the entity.

Add mobile device features (Photo, Gallery, Phone, Mail, SMS, Contact, Maps, Barcode)

The default template for modeling mobile apps includes features for interacting with a mobile device's default apps such as phone, email, text, photos, and maps.

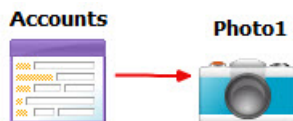
The process for adding mobile device features to a BA diagram is the same for each mobile device feature, but each feature has specific properties to set on the relation to generate the appropriate code.

This task assumes that you have created a [BA diagram](#) with a CRUD or Zoom Form entity.

1. Right-click on the BA diagram and select **New**. Select a mobile device feature to add to your form.

Option	Description
Photo	Lets the user take a picture with the mobile device and returns the corresponding picture identifier.
Gallery	Lets the user select a picture from the mobile device's photo gallery and returns a picture identifier.
Phone	Calls the selected telephone number.
Mail	Invokes the user's default mail application for a new mail to send.
SMS	Sends an SMS text to one or more phone numbers.
Contact	Lets the user choose a contact from the mobile device contact list and returns the vCard.
Maps	Invokes a maps app with the current Global Positioning System (GPS) location of the mobile device.
Barcode	Lets the user to scan a barcode with a mobile device and returns the string representation of the barcode and its type.

2. Right-click the form icon and select **New Relation**. Click the Form icon again and drag the relation arrow to the new mobile device feature icon (such as **Photo**).



3. Select the relation.
4. Set the `action` property. The `action` property on the relation must match the `action` property on the form widget used to invoke the feature. Set any other properties specific to the mobile device feature. See [Mobile device function properties](#) on page 223.
5. Build and run your program.

Mobile device function properties

Relation properties specific to mobile device feature entities.

Note: For all mobile feature relations, the `Action` property value on the relation must match the `Action` property value on the form widget used to invoke the feature.

Photo / PhotoRelation

Lets the user take a picture with the mobile device and returns the corresponding picture identifier.

Table 45: PhotoRelation properties

Property	Usage
Image Path	Column to store the picture identifier.

Gallery / GalleryRelation

Lets the user select a picture from the mobile device's photo gallery and returns a picture identifier.

Table 46: GalleryRelation properties

Property	Usage
Image Path	Column to store the picture identifier.

Phone / PhoneRelation

Calls the selected telephone number.

Table 47: PhoneRelation properties

Property	Usage
Phone Number	Column corresponding to the telephone number.

Mail / MailRelation

Invokes the user's default mail application for a new mail to send.

Table 48: MailRelation properties

Property	Usage
To	Columns corresponding to the To line of the email.
CC	Columns corresponding to the CC line of the email.
Bcc	Columns corresponding to the Bcc line of the email.
Subject	Column corresponding to the subject line of the email.
Content	Column corresponding to the body of the email.
Attachment	Columns corresponding to email attachments. Limit 2.

SMS / SMSRelation

Sends an SMS text to one or more phone numbers.

Table 49: SMSRelation properties

Property	Usage
To	Column corresponding to the SMS number.
Content	Column corresponding to the body of the text.

Contact / ContactRelation

Lets the user choose a contact from the mobile device contact list and returns the vCard.

Table 50: ContactRelation properties

Property	Usage
Vcard	Column corresponding to returned Vcard value.
Person	Columns corresponding to the contact.
Address Work	Columns corresponding to the contact's work address.
Address Home	Columns corresponding to the contact's home address.
Phone	Columns corresponding to the contact's phone numbers.
Email	Columns corresponding to the contact's email addresses.

Barcode / BarcodeRelation

Lets the user to scan a barcode with a mobile device and returns the string representation of the barcode and its type.

Table 51: BarcodeRelation properties

Property	Usage
Barcode	Column corresponding to the barcode string value.

Map / MapRelation

Passes latitude and longitude values to the mobile device map utility.

Table 52: MapRelation properties

Property	Usage
Query	Columns corresponding to the latitude and longitude values.

Import files into the diagram from the project

If there are files in the project structure that are not in the Business Application diagram, they can be imported.

Right-click on the background of the diagram, and select **Import Project Files to BA Diagram**.

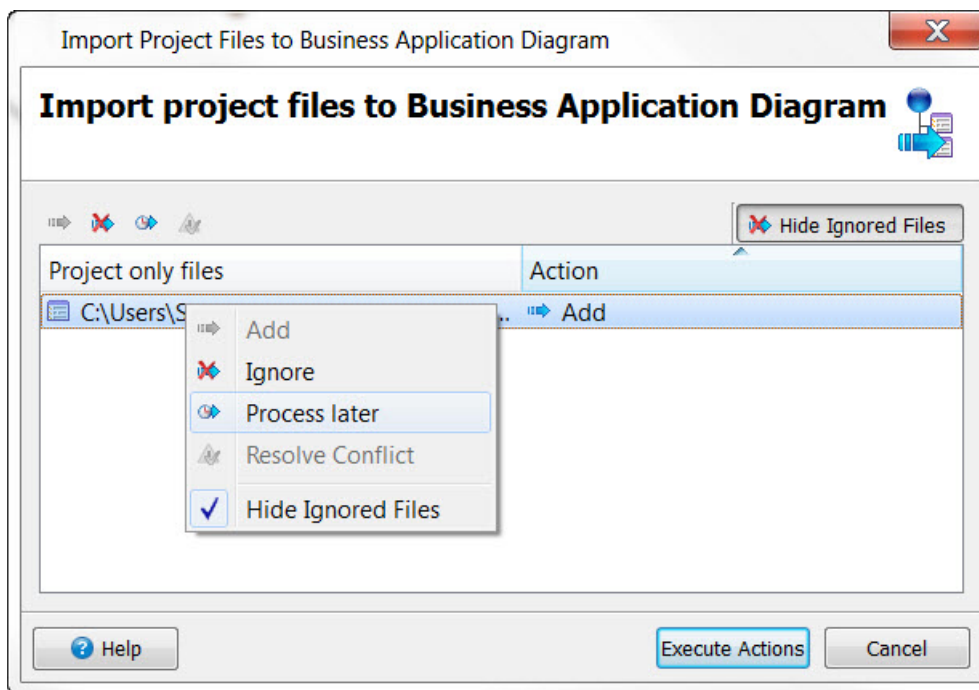


Figure 157: Import project files to Business Application diagram dialog

Files that can be imported are listed in the dialog. Select a file, and choose an option from the integrated Toolbar or right-click context menu.

Ignore	Do not add selected files to the diagram.
Process later	Decide later if the selected files should be added to the diagram.
Add	Add selected files to the diagram.
Resolve conflict	Resolve an existing conflict.

Modeling the database

Model the database by creating a meta-schema or extracting one from an existing database.

Warning: A rebuild of a project is not automatically done when the meta-schema file (.4db, .4dbx) is modified. It is the responsibility of the developer to recompile the appropriate parts of the project.

- [Database meta-schema \(4dbx\)](#) on page 226
- [Create a meta-schema](#) on page 227
- [Extract meta-schema information from database](#) on page 228
- [Add a meta-schema to a project](#) on page 231
- [Managing SERIALs in a generated application](#) on page 231
- [Managing concurrency](#) on page 231
- [Cascade delete](#) on page 232

Database meta-schema (4dbx)

The 4dbx file is the database schema file for generated applications.

The 4dbx file contains the information about the tables, columns, and relations of the relational database that is needed by your application. It is used to create items in your Business Application diagram (4ba).

The 4dbx file handles CRUD operations for the generated application. For each table with a primary key in the database schema, the application generator creates its database CRUD operations (respectively INSERT, SELECT, UPDATE, and DELETE database statements).

You must have [access set up for the database](#) that the application will use.

Constraints management

The Application Generator creates functions to manage the database schema constraints. These constraints are checked during the CRUD operations for each table:

- uniqueness check for its primary key (only if a primary key exists)
- uniqueness check for each unique constraint (only if a primary key exists)
- for one column, the not null constraint is checked
- foreign key existence is checked

SERIAL management

You can define a primary key column as a SERIAL to use sequence numbers for the keys. BAM manages the SERIAL columns by using a single table named seqreg in the meta-schema (4dbx).

Create a meta-schema

Create a database meta-schema file.

1. Select **File >> New**.
The **New** dialog opens.
2. Select a category from the left-side panel.

Option

Genero BAM Desktop

Description

Select this option if you are creating a database meta-schema for use in a Business Application Modeling managed project for a desktop application. The file created will be a 4dbx.

Genero BAM Mobile

Select this option if you are creating a database meta-schema for use in a Business Application Modeling managed project for a mobile application. The file created will be a 4dbx.

Genero

Select this option if you are creating a database meta-schema for use in a standard Genero program. The file created will be a 4db.

3. Select an option from the Database section from the right-side panel.

Option

DB Schema

Description

Creates a new meta-schema file.

DB Schema from Database

Opens the **New Meta-schema** dialog to which you enter your database and connection information. See [Extract meta-schema information from database](#) on page 228.

The meta-schema file displays in the document view. A new meta-schema file is automatically added to the list in the [DB Schemas Tab](#) if you have saved the file in the current project.

Extract meta-schema information from database

The **New Meta-schema** dialog assists in extracting schema information from a database.

To extract a database meta-schema file from a database, you must have access and permissions for the database. If you have trouble connecting to a database, make sure the database and the corresponding database client software are installed and configured properly.

When you extract the meta-schema information from the database, you overwrite the existing schema. Any user changes that had been made to the schema are lost when using the extract schema option. If you wish to keep user changes, you must update the schema. See [Update a meta-schema from database](#) on page 302.

1. Select **Database>>Extract Schema**. The first step is specifying the name and location of the meta-schema file.

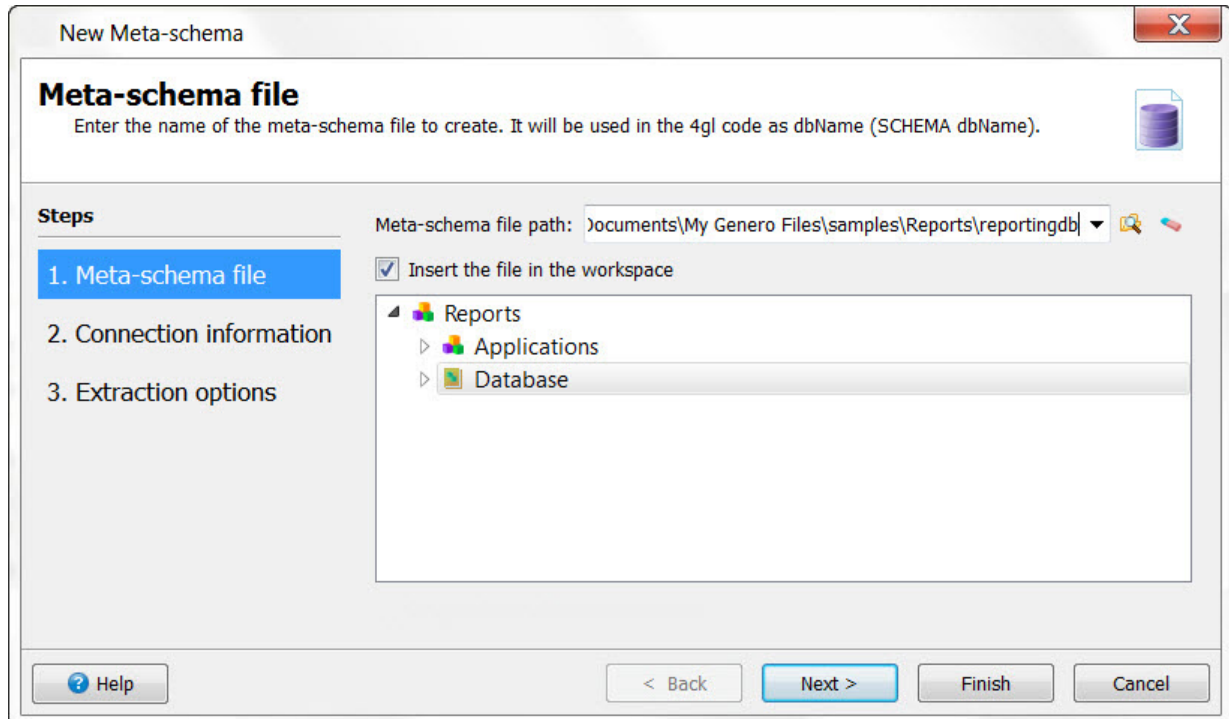


Figure 158: New Schema dialog

Meta-schema file path

Browse for a destination directory. Enter the name and path for the new database meta-schema file. Select a 4db meta-schema file for standard projects. Select a 4dbx file if you are working with a Business Application Modeling managed project.

Insert the file in the project

Check this box to add the meta-schema file in the project. Select the node where the file should be added.

2. Click the **Next** button to continue to **Connection information**. This connection information is only used to extract the information for the database meta-schema file from the referenced database.

New Meta-schema

Connection information
Enter the connection information required to connect to the database.

Steps

1. Meta-schema file
2. Connection information
3. Extraction options

Database Connection Information

Use explicit settings :

Previous connections :

Database type :

Database driver :

Informix server :

Database name :

Use external settings :

Extraction process relies on FGLPROFILE information to connect to the database. The provided schema name is used to connect to the database. Please refer to the Genero BDL manual for more information.

Database User Information

Table owner :

User name :

User password :

Figure 159: Connection information

- a) In the **Database Connection Information** section, select either **Use explicit settings** or **Use external settings**.

Use explicit settings, previous connection

You can use a **previous connection** that was created for the same database. The drop down list provides a list of the existing connections.

Use explicit settings, database type

You can enter the **Database Type** by selecting the desired type from the drop down list, and the corresponding information for that type. The **Database driver** for the database type is automatically entered. If other drivers exist, they are available in the drop down list.

Use external settings

Information in the `FGLPROFILE` configuration file is used to extract the corresponding connection information for the specified database. Genero Studio will use the schema name that you entered to check for any related entry in the `FGLPROFILE` configuration file, and will use those values to define the connection. See information on the `FGLPROFILE` file in the *BDL User Guide*.

- b) In the **Database User Information** section, provide the necessary database user details. The required information varies based on the database type selected. See [Database server/user information](#) on page 308.
- c) Click **Test Connection** to verify that the information is correct and that you are able to access the database.
3. Click the **Next** button to continue to **Extraction Options**. Select the options for the meta-schema file.

New Meta-schema

Extraction options
Select the extraction options you wish to use.

Steps

1. Meta-schema file
2. Connection information
3. Extraction options

Options

Case sensitivity :

Import system tables

Ignore errors

Conversion Method

Base type	Type A	Type B
BIGINT	<input checked="" type="checkbox"/> BIGINT	<input type="checkbox"/> DECIMAL(19,0)
BIGSERIAL	<input checked="" type="checkbox"/> BIGSERIAL	<input type="checkbox"/> DECIMAL(19,0)
BOOLEAN	<input checked="" type="checkbox"/> BOOLEAN (t=45)	<input type="checkbox"/> CHAR(1)
INT8	<input checked="" type="checkbox"/> INT8	<input type="checkbox"/> DECIMAL(19,0)
LVARCHAR(m)	<input checked="" type="checkbox"/> VARCHAR2(m)	<input type="checkbox"/> VARCHAR2(m)
SERIAL8	<input checked="" type="checkbox"/> SERIAL8	<input type="checkbox"/> DECIMAL(19,0)

Buttons: ? Help, < Back, Next >, Finish, Cancel

Figure 160: Extraction options

Case sensitivity

Specify how case in database object names should be handled. **Case sensitive**: case won't be changed on database objects, **Lower case**: database object names will be converted to lower case, **Upper case** : database object names will be converted to upper case.

Import system tables

Check this box to include system tables in the schema.

Ignore errors

Specify that conversion errors should be ignored. If this option is unchecked, the extraction will stop as soon as an error occurs (for example, if a table column has an unsupported type.)

Conversion method

Select the type of conversion you wish for the specific data types; the default choice is Type A.

4. Click **Finish** to begin the extraction process.

If you didn't already do so, save the database meta-schema file in a node in the project; the database meta-schema will be added to the DB Schemas tab and made available to other modules.

Any application that uses the meta-schema file must have a dependency to the node where the meta-schema file was added. See [Add a meta-schema to a project](#) on page 231.

Add a meta-schema to a project

You may have to add the meta-schema file to a project.

If you used **File >> New** to create your project, the default structure of your project includes nodes for a **project**, **application**, **library**, and **databases**; the dependencies between the default nodes has been predefined. When you save your Meta-schema file in the Databases node of the project, the dependency for the application node in the project already exists.

However, if you created your own project structure, you must follow these steps.

1. Open the project.
2. Right-click on the application or library node in the Project view to which you want to add the meta-schema file and select **Add Files**. Locate and add the [meta-schema file](#).
3. Add a dependency to the Meta-schema file for any application or library nodes. Right-click the node and select [Advanced Properties, Dependencies](#). Check the box for the node containing the Meta-schema file.

Managing SERIALs in a generated application

You can define a primary key column as a `SERIAL` to use sequence numbers for the keys. BAM manages the `SERIAL` columns by using a single table named `seqreg` in the meta-schema (4dbx).

The seqreg table

For generated applications, BAM automatically adds a `seqreg` table to a new meta-schema (4dbx) to facilitate sequence number management for `SERIAL` columns. The `seqreg` table contains two columns; one with the name of the table having a `SERIAL` column and one with the last `SERIAL` column value for that table.

No configuration required

Once you have created a table with a `SERIAL` defined as the primary key, BAM manages sequence number generation and generates all code necessary to handle inserting the new sequence number during an Add operation. No addition configuration or modeling is required.

You may initialize the `seqreg` table with the appropriate records otherwise `SERIAL` columns will start at number 1.

If the database schema does not contain `SERIAL` columns, the `seqreg` table is not required in the schema.

Informix databases

The `seqreg` table is required in a schema containing `SERIAL` columns except for Informix® databases where the native Informix® `SERIAL` management is used.

For further information on how the `seqreg` table is used, see the *BDL User Guide, "Solution 2: Generate serial numbers from your own sequence table"*.

Managing concurrency

The default template manages concurrent access.

Concurrent access occurs when two or more users work on the same data, such as when more than two users are working with the same application or with different applications using common data.

Note: With mobile apps, only one user accessing the app at any given time; data cannot be altered by another user. Concurrent access management is disabled for generated mobile apps.

The default template checks for concurrent access and notifies the user that the data being updated has been *changed* by another user (either deleted or updated). The concurrent access only applies to data of the current row. The check of the concurrent access is systematically done when the user:

- starts to modify the current row
- saves updates to the current row
- deletes the current row

If the user's data was not updated by another user the application follows its normal course, such as insert, update or delete of the current row.

If the user's data was updated by another user, a warning is raised and the user is asked to refresh the data. If the user rejects the request to refresh the data, the application stays in its current state and the application's process does not continue. If the user accepts the request to refresh the data, the current row is refreshed.

If the user's data was deleted by another user a warning is raised and the user is asked to refresh the data. If the user rejects the request to refresh the data, the application stays in its current state and the application's process does not continue. If the user accepts the request to refresh the data, the current row is deleted from the current data set.

Concurrent access management uses the optimistic locking strategy. The data that the user wants to update or delete is compared to what is stored in the database ensuring no change has occurred. A check is done when starting to modify the current row in the application to detect data modification at the earliest. See the topic **Optimistic Locking** in the *Genero Business Development Language User Guide* for more information.

Note: In the database layer, the default template uses physical row locking for update and delete operations. This ensures data cannot be changed during the database transaction. Physical row locking can only be done with databases that handle locks.

Cascade delete

A foreign key with a cascade delete specifies that if a row in the parent table is deleted, then the corresponding rows in the child tables are automatically deleted.

Note: An infinite delete cascade cycle can occur when table A references table B, table B references table C, table C references table A (and so on) and all delete cascade boxes are checked. While running the delete cascades, if a row is found that was a candidate for deletion during the cycle, an infinite delete cascade cycle occurs and results in a runtime error.

Working with forms

Special properties are used to specify what logic should be generated for a form and how the form should behave during the various states (DISPLAY, MODIFY, ADD, SEARCH).

- [Mobile forms](#) on page 233
- [Enable and disable CRUD logic](#) on page 239
- [Form behavior in CRUD states](#) on page 241
- [Control the row position in form](#) on page 244
- [Opening a form with a subset of data](#) on page 244
- [Field activation](#) on page 244
- [Define queries and data order](#) on page 245
- [Define a dynamically populated ComboBox](#) on page 245
- [Lookup fields](#) on page 247
- [Add buttons to form](#) on page 249

- [Add formonly \(nondatabase\) fields to a form](#) on page 250
- [Master-detail forms](#) on page 250

Mobile forms

This section includes topics on working with forms for mobile apps.

- [Mobile form patterns](#) on page 233
- [Display image with table row in mobile form](#) on page 235
- [Reuse a common form](#) on page 237
- [Rowbound actions](#) on page 237

Mobile form patterns

Mobile platforms require special form patterns. These patterns can be modeled in BAM.

Note: Examples shown here can be found in the **MobilePatterns** demo from the Welcome Page.

Example: Table with 2 actions opens a common CRUD form

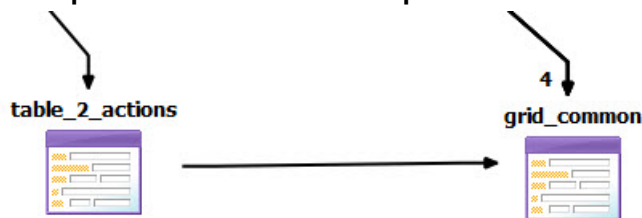


Figure 161: Model of table to grid form pattern

This common mobile pattern can be modeled in BAM by creating a table container form to display the rows and setting a relation to a grid container form for CRUD operations on the individual record selected from the table display.

To see this example model, open the **MobilePatterns** project from the Welcome Page and open the `MobilePatternsAppFlow.4ba` file. To run this example, execute **MobilePatternsApp** to your emulator or device. Select **Table Usage >> 2 actions on the Table**.

On mobile platforms, a `table` container displays as a *list view*, not as a grid with columns. As a list view, it only displays the two first columns' content and any associated row image. There is no way to manipulate columns (hide, reorder, resize, or sort).

Depending on the alignment property of the two columns, a row is displayed as either

- On two lines for each row.
- On a single line with the first column is left-aligned and the second column is right-aligned. For example, if the first column is a text column (left-aligned) and the second column is a number (right-aligned), they will be rendered on a single line for the row.

If a table is the one and only element in a form, a standard table view displays.

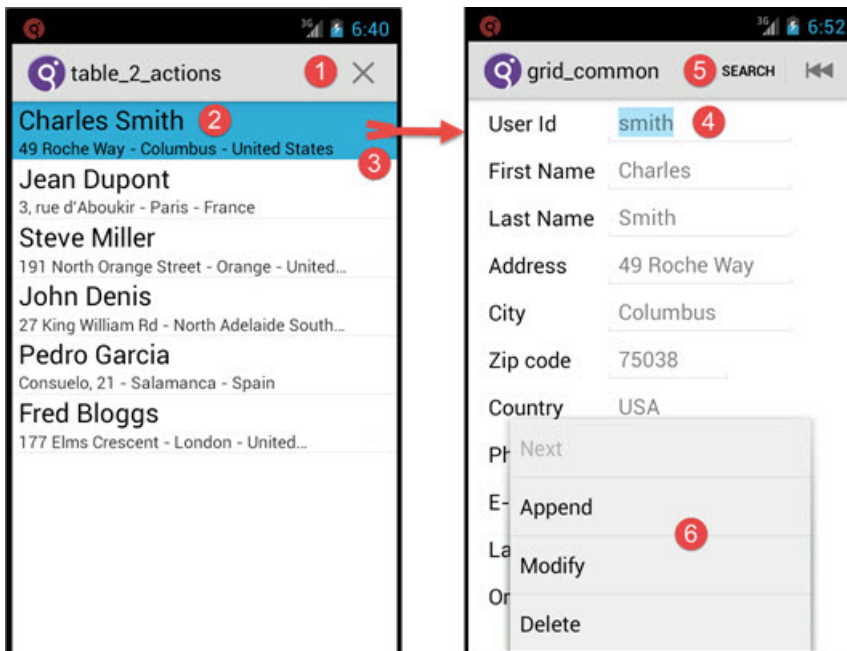


Figure 162: Running app with table form to grid form pattern

1. The **table_2_actions** form includes a table that has only the `delete` action code generated. (The record for this form specifies only `canDelete` functionality.) The icon or a swipe on the row triggers the `delete` action.
2. This table displays rows with two lines on each row. The first column is displayed above the second column. In the table container definition, two display only columns have been added to the table to concatenate the first and last name data in the first position and the address data in the second position. The logic to concatenate the data for these fields has been added as custom code in a `POINT`. See [Finding the right place to customize](#) on page 259. For example:

```
{<POINT Name="fct.record1_computeFields.user" Status="MODIFIED">}
  LET l_rec_BRComputedFields.recordfield1 = SFMT("%1 %2",
    p_br_fields.account_firstname
    CLIPPED,
    p_br_fields.account_lastname
    CLIPPED)
  LET l_rec_BRComputedFields.recordfield2 = SFMT("%1 - %2 - %3",
    p_br_fields.account_addr1
    CLIPPED,
    p_br_fields.account_city
    CLIPPED,
    p_br_fields.country_codedesc
    CLIPPED)
{</POINT>}
```

3. The user can browse the list, delete a selected row, or tap the row to go to a form populated with the selected record. The `doubleclick` property on the table specifies the action triggered when the row is selected. In this example the value of the `doubleclick` action is `open_common_form`. The code is automatically generated to open the form connected to the table form by a relation in the model.

See the *Tables* topic in the *Genero Mobile Developer Guide* for more on table rendering and ergonomics in mobile apps.

4. The form opens and displays the correct record. This is achieved by setting some properties on the relation to the grid form on the BA diagram:
 - `Position`, `Source Field` property is set to `account.userid`.
 - `UI Settings`, `Action` is set to the `open_common_form` action.

- UI Settings, Open Mode is set to **DISPLAY**.
5. All CRUD functionality is generated for the grid_common form (search (QBE), read and delete, create, and update). Available actions display in the order the ON ACTION statements appear in this dialog in the code. Only the first few actions appear in the menu bar at the top, depending on mobile device type.
 6. All other available actions appear when the full menu is displayed by the user (by pressing a menu button on a phone, for example).

Example: Single CRUD forms or Common CRUD form

There are two options when modeling CRUD grid forms for mobile. Typically a grid form is used for all CRUD operations. You can model this by creating one common form and managing which operation(s) are available through the relation to the common form.

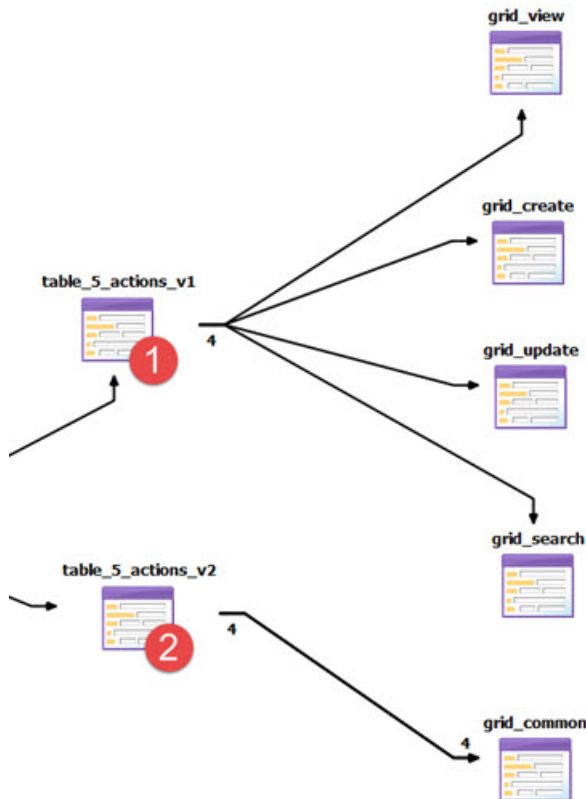


Figure 163: Many vs one common form

1. The **table_5_actions_v1** table form calls a different form depending on the action the user triggers. Each form handles a single operation such as (search (QBE), read and delete, create, and update).
2. The **table_5_actions_v2** table form calls just one form, but models four different relations to the common form. Each relation opens the form for a different operation such as (search (QBE), read and delete, create, and update). The **Open Mode** and **Action** properties are uniquely set on each of the four relations. For example, one of the four relations has the **Action** property set to **append** and the **Open Mode** property set to **ADD**. When the user triggers the **append** action, the common form is opened and ready to accept input. Furthermore, the **ADD Behavior** properties on the relation overwrite the form defaults and specify to **ExitForm** when the user accepts or cancels the input. See [Overwriting a Form's behavior with a Relation to the Form](#) on page 242

Display image with table row in mobile form

You can display an image with each table row in a mobile form.

Note: Example shown here is the **OrderLines** form in the **OfficeStoreMobile** demo.

Example: Table with image

This app displays a form that is made up of a table container. The table container has special features for mobile apps.

1. An image in the first position.
2. Concatenated data displayed in two fields. See [Mobile form patterns](#) on page 233.

In this example, the database table includes a column (prodpic) that contains the name of the image to display for the selected row. During development, the images are found in the directory specified by the FGLIMAGEPATH environment variable. For apps that will be deployed, the packaging specifies where to find the images for inclusion in the app package.

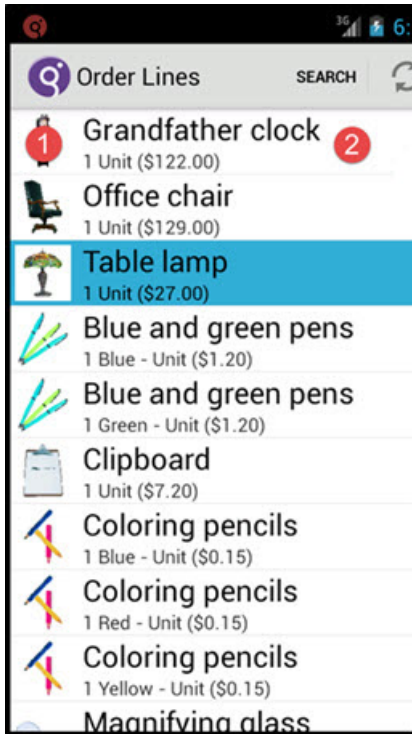


Figure 164: Running app with table form with image

Form structure

To create this table view for your mobile app:

- The table structure for this form includes a picture column set to a `Phantom` type, and two fields (`masterInfo` and `detailInfo`) to hold the concatenated display data.
- The `masterInfo` field has the `IMAGECOLUMN` attribute (seen as the **image column** property in Genero Studio) set to the name of the field containing the name of the image.

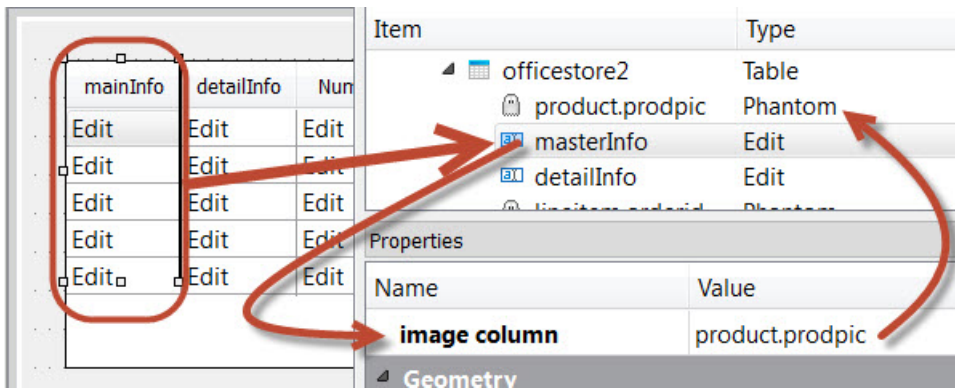


Figure 165: masterInfo column with image column property set

When rendered in a mobile environment, the image defined by the `Phantom` field displays in the first position of each row in the table.

See also *Tables* in the *UI Behavior* section of the *Genero Mobile User Guide*.

Reuse a common form

Writing applications often requires creating a lot of small forms, especially for mobile apps. To make app management easier, consider creating a common form that has all operations generated for it, then limiting the operations as it is opened in different scenarios in the app.

There are several ways to reuse a form in an application. For an example of setting many relations to a common form, see [Example: Single CRUD forms or Common CRUD form](#) on page 235.

Use it as is

In the BA diagram, set a relation to the form as is.

Reuse a subset of functionality

Create the form with all CRUD operations generated. Then, in the BA diagram relation to the form, set the properties to **disable Add**, **disable Modify**, **disable Delete**, and/or **disable Search**. When the form is opened, only a subset of the operations will be available to the user.

Display a subset of the data depending on the parent form

The **Filter** properties on a relation are used to apply a SQL filter to show a subset of data when opening a form. See [Opening a form with a subset of data](#) on page 244

Position the cursor depending on the parent form

When a form is opened from another form, the current position is always set to the first row. With positioning, you can control the row that is preselected when the form is opened. The **Position**, **Source Field** property on the relation to the form controls which row will be displayed when the form is opened. See [Control the row position in form](#) on page 244.

Rowbound actions

Set the **Row Bound** property on a relation between forms to render a contextual action to the current row.

The **Row Bound** property specifies whether the action set in the **Action** property will appear as a contextual menu option on the row. If your form only has one record, BAM automatically uses that record as the **Source Record** for the **Row Bound** property. If your form has more than one record, you must set the **Row Bound** *and* the **Source Record** property to specify for which record the **Row Bound** property is to be applied.

To set an action other than the action being used to open a form to rowbound, customize the action in the code with `ATTRIBUTES(ROWBOUND)`. For example, this custom code adds a user action `print` and makes it rowbound.

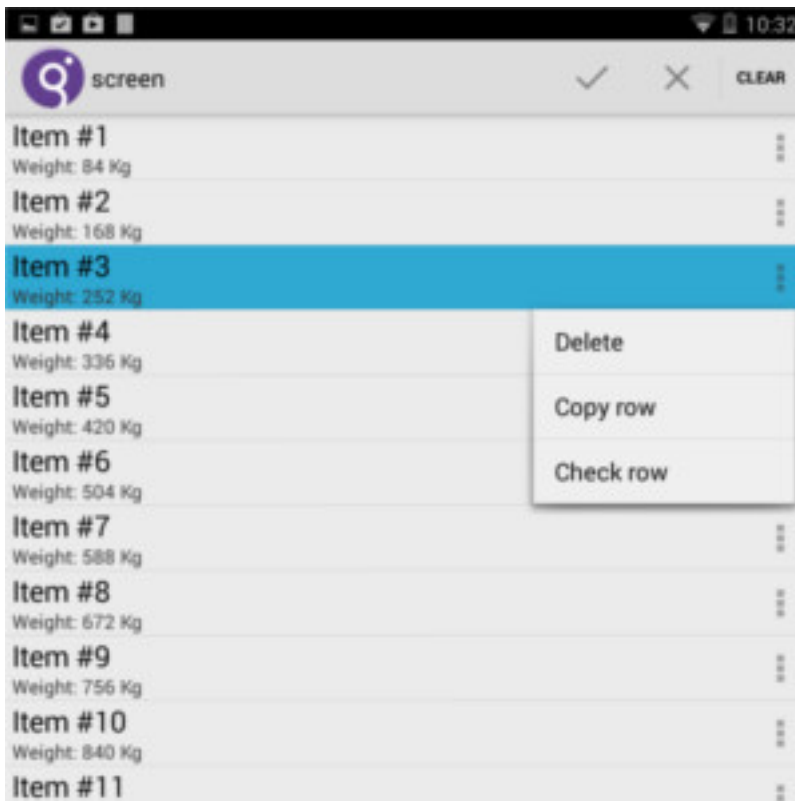
```
{<POINT Name="dlg.recOrders.uiDisplay.userControlBlocks"
Status="MODIFIED">}
  ON ACTION print ATTRIBUTES(TEXT = "Print", ROWBOUND)
    DISPLAY "print"
{</POINT>}
```

The `delete` action is always available via swipe left unless you have set the `canDelete` form functionality property to unchecked (false). See [Enable and disable CRUD logic](#) on page 239.

For more information on rowbound actions, see the *Rowbound* topics in the *BDL User Guide*.

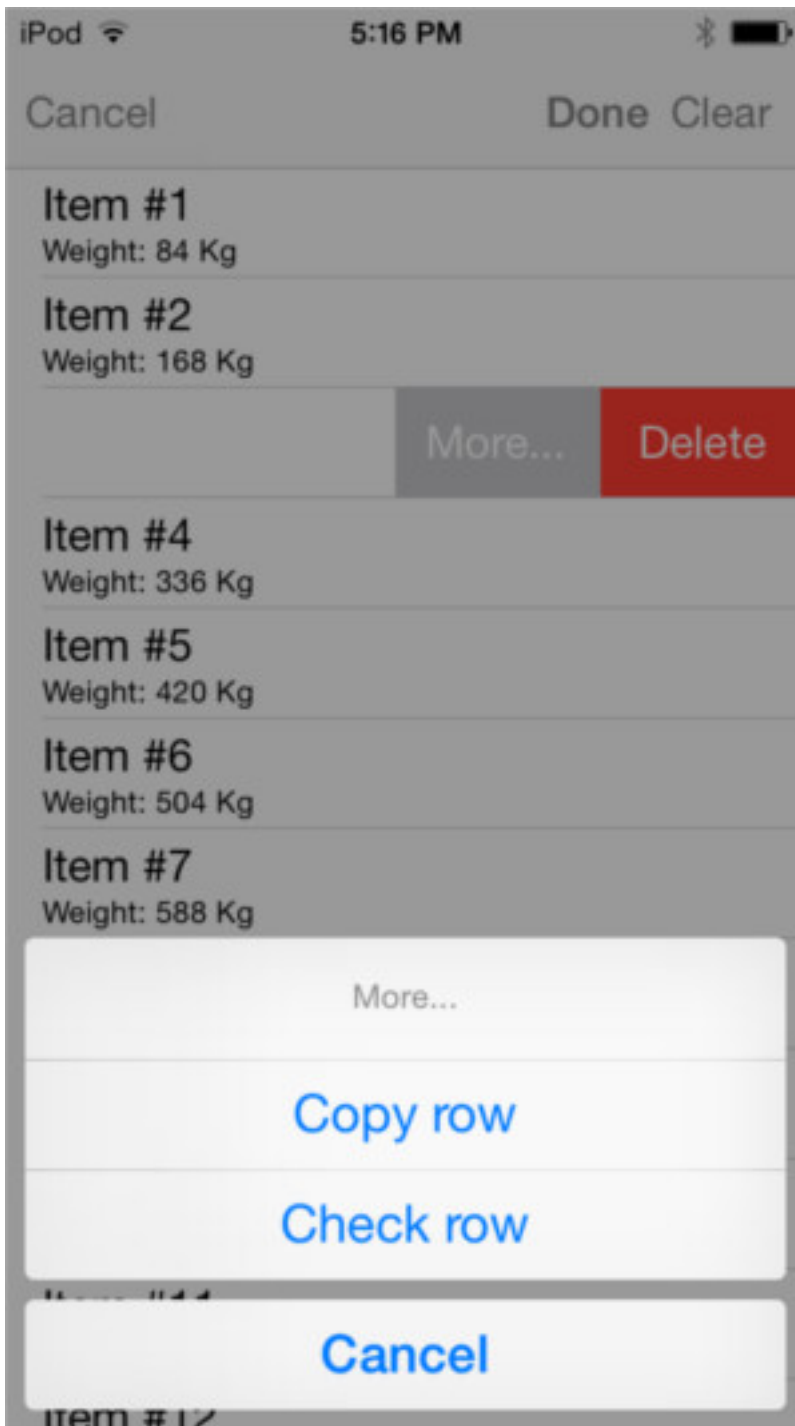
Android list view with rowbound actions

On Android, when rowbound actions are defined, each row of a list view shows the three-dot indicator. Tap this icon to bring up a row context menu with options to execute the corresponding rowbound actions. Swipe the row from the right to the left to fire the delete action, if defined.



iOS list view with rowbound actions

On iOS devices, when you swipe your finger from right to left, **More...** and/or **Delete** icons show up in the row. Tap **More...** to bring up a list of rowbound actions to execute. Tap **Delete** to fire the corresponding delete action code.



Enable and disable CRUD logic

Form functionality properties (`canDisplay`, `canAdd`, `canModify`, `canDelete`, `canSearch`, `canEmpty`) can be set on each record of a form to specify whether the program logic of display, add, update, delete, search and/or display empty should be generated. Generated functionality can be disabled on a relation to the form in the BA diagram.

Functionality properties on the form record

The program and user interface logic is generated when functionality properties are set. The state of the action (enabled/disabled) or the availability of the action in the Toolbar and/or Topmenu depends on the

setting of the **Functionality** properties. For example, if the `canSearch` property is checked, the form will allow for data queries and the Toolbar and Topmenu will include a button and menu option for searching.

canDisplay	This form can be used to display data in this record. Allows the user to browse records with next and previous.
canAdd	The form can be used to add data in this record.
canModify	The form can be used to modify data in this record.
canDelete	The form can be used to delete data in this record.
canSearch	The form can be used to search data in this record. <i>canSearch</i> is the only functionality property available to Zoom forms.
canEmpty	This form can be used to sit empty waiting for the user to trigger an action. Useful when a form is to be presented empty, prior to running a search query.

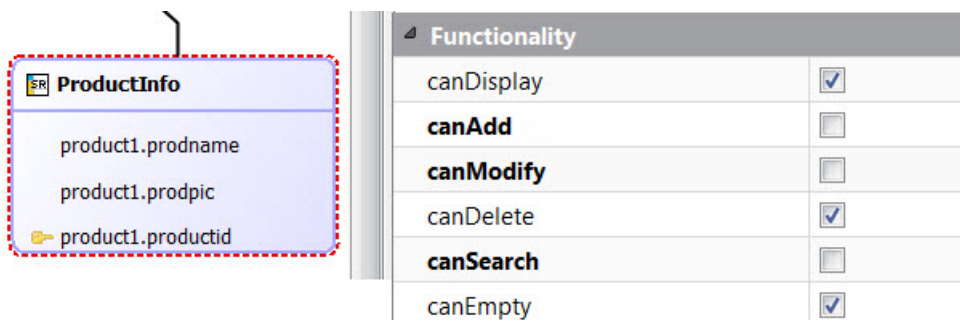


Figure 166: Setting the record Functionality properties

Disabling Functionality properties on a relation to a form

You can disable functionality generated for a form in a relation to the form on the BA diagram. For example you may have a form that generates code for all functionalities, but in some situation when the form is opened you do not want a functionality to be available. In this example, the form record has all functionalities selected, thus all CRUD logic will be generated. On a specific relation to this form, however, the Add and Search functionalities are disabled preventing the user from adding or searching from this form when opened.

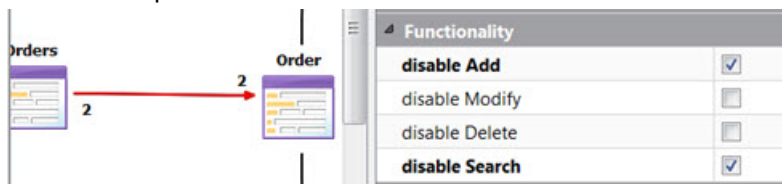


Figure 167: Disabling some of the generated functionality on the relation to the form

Form behavior in CRUD states

The **UI Settings** properties specify the initial and default state of the selected form as well as the behavior of the form during each state (DISPLAY, MODIFY, ADD, and SEARCH). These settings can be set on the form entity in the BA diagram, but overwritten by a relation to the form.

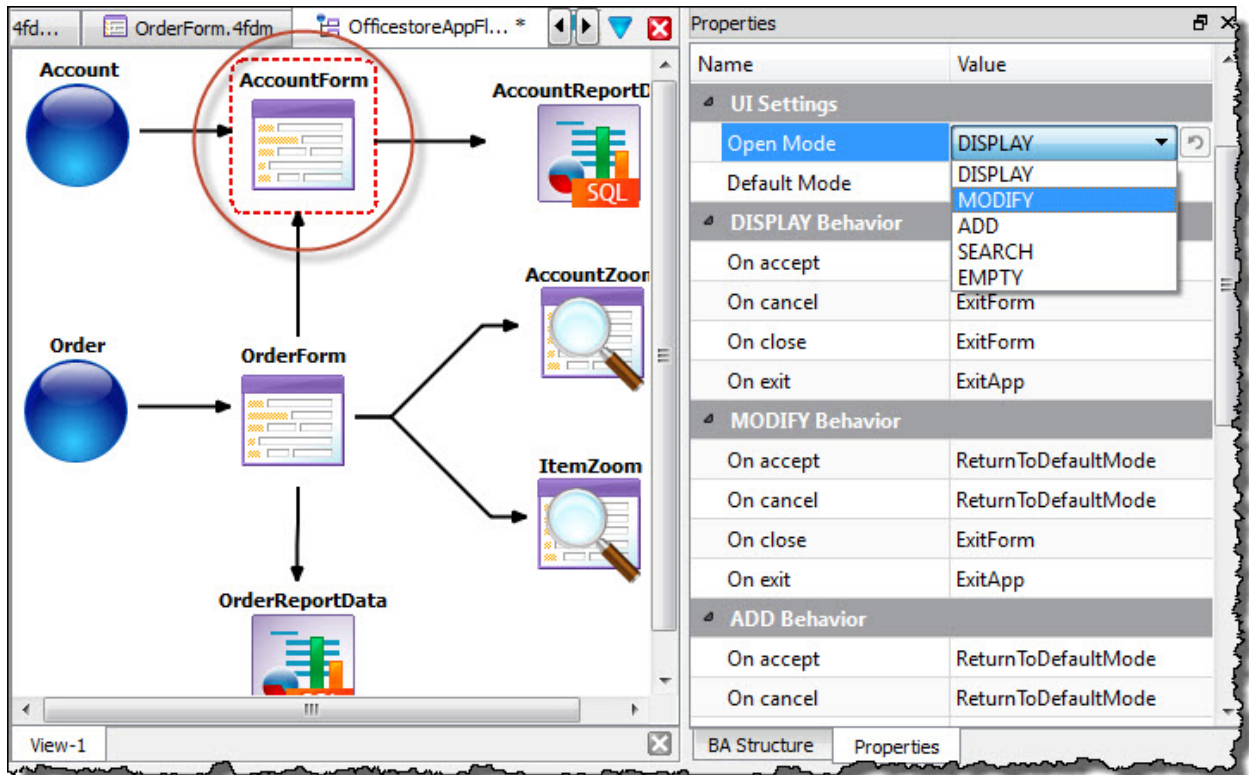


Figure 168: UI Setting properties

Initial and default states

Open Mode

Open Mode is the initial state of the form when opened. The rendered form's default Toolbar allows the user to switch modes.

Default Mode

Default Mode is the mode in which you return after leaving another mode.

Table 53: Form States

Option	Description
DISPLAY	The default mode. Data is retrieved from the database and displayed as a record on the form, one at a time. The default Toolbar allows the user to scroll through the list of records.
MODIFY	The user can modify the currently displayed record.
ADD	The user can enter a new record.
SEARCH	The user can enter criteria for a search of the database, displaying the records that match that criteria.

Option	Description
EMPTY	An empty form is displayed. The user can select the search or add actions from the Toolbar to change the mode.

Behavior properties

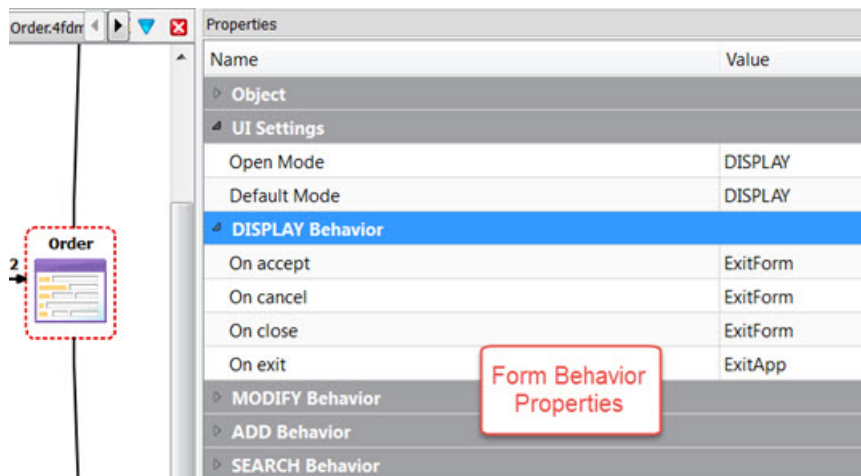


Figure 169: Form Behavior properties

The **Behavior** properties control how the form should behave during each state: `DISPLAY`, `MODIFY`, `ADD`, `SEARCH` for CRUD forms and `DISPLAY`, `SEARCH` only for Zoom forms.

On accept	What should happen when the <code>accept</code> action is triggered.
On cancel	What should happen when the <code>cancel</code> action is triggered.
On close	What should happen when <code>close</code> action is triggered.
On exit	What should happen when the <code>exit</code> action is triggered.

Table 54: Behavior properties

Option	Description
ExitForm	Close the form and close the application if it is the last form open.
ExitApp	Close the application.
ReturnToDefaultMode	Return to the mode specified as the default in the Default Mode property.
ReturnToCallerMode	Return to the previous mode.
StayInMode	Stay in the current mode.

Overwriting a Form's behavior with a Relation to the Form

Property values on a relation to a form overwrite the form's default property values. This allows you to set default property values on the form, but change them on the relation to the form.

In the modeling of mobile apps, it is common to have multiple relations to a common form. This allows you to reuse a common form, opening it in a different state depending on the action triggered. In this example, there are 2 relations to the `Order` form. Each relation specifies how the form should behave depending on the action that opens it. One of those relations, `Add an Order`, specifies that when the form is opened with the `append` action, the form opens in the `ADD` state. It also changes the default `ADD` behavior by exiting the form (`ExitForm`) when the user either accepts or cancels the `ADD` operation.

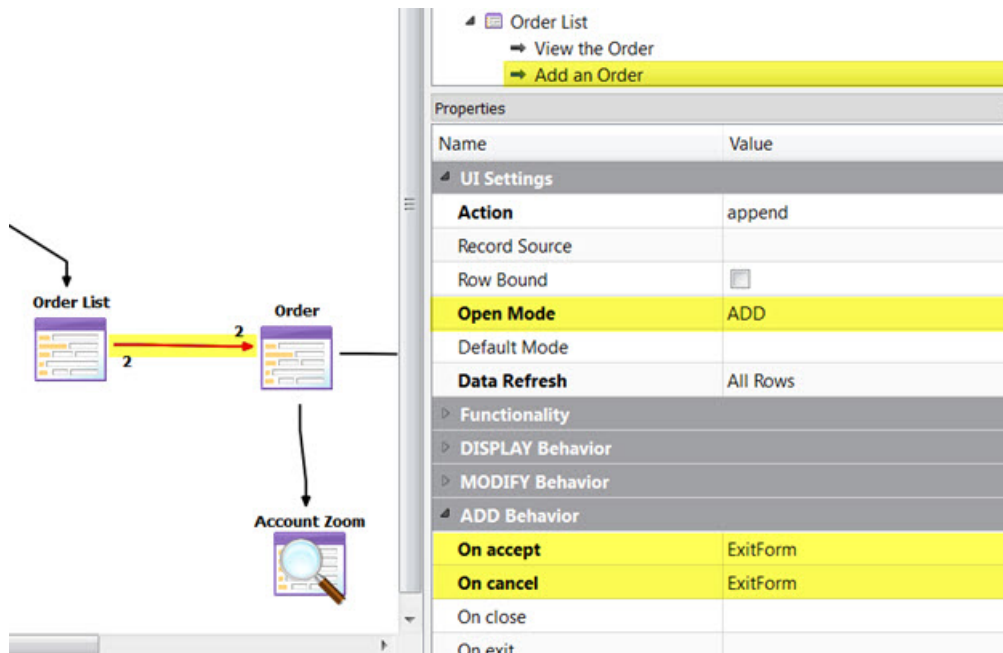


Figure 170: Different Relations to a Common Form

Data refresh

Data refresh behavior is specified with the `Data Refresh` property on the relation between two forms. If the source form is in `DISPLAY` mode and opens a destination form, when the destination form closes, the data displayed in the source form refreshes.

Table 55: Data Refresh property values

Option	Description
All Documents	All data are refreshed.
Current Document	The current document (current row of the master record) is refreshed. This is the default value.
All Rows	All rows of the current record defined by the <code>Position</code> , <code>Source Field</code> property values are refreshed.
Current Row	The current row defined by the <code>Position</code> , <code>Source Field</code> property values is refreshed.
None	No refresh is done.

The **Current Row** and the **Current Record** are defined by the `Position`, `Source Field` properties of the relation. If the `Position`, `Source Field` property values are the unique key of the record, then the `Data Refresh` uses the row defined by the returned values.

If the `Data Refresh` is for **All** or **Current Document**, then the `Data Refresh` uses the current document by following the record relations from the row of the record defined by the `Position`, `Source Field` property values to the master record (from unique key to unique key of the parent).

Note: If the `canAdd` property is enabled in the destination form, set the `dataRefresh` property to **All Rows** or **All Documents**, otherwise the added rows will not be displayed on return to the source form.

Control the row position in form

When a form is opened from another form, the current position is always set to the first row. With positioning, you can control the row that is preselected when the form is opened. The `Position`, `Source Field` property on the relation to the form controls which row will be displayed when the form is opened.

There are several ways that you may want to control the row position when a form is opened. The `Source Field` must match the target form field in number and type for the positioning to work.

Open to the same record as the parent form record

Set the `Source Field` property to the primary key of the parent form record. When the form opens, it will open to the same record as the parent form.

Open to a position depending on a field value of the parent record

Set the `Source Field` to the field of the parent record you want to use. For example, an account form and a country form may have a relation between them that specifies the `account.country_id` as the positioning `Source Field`. While in the account form, suppose the user is on a record with a country field value of "USA". If the user then opens the country form from the account form (through a Zoom form, for example), the `country_id` ("USA") will be preselected in the form instead of the first record. Often this will be a field that corresponds to a foreign key. (For example, the `account` table has a foreign key to the `country` table.)

Opening a form with a subset of data

The `Filter` properties on a relation are used to apply a SQL filter to show a subset of data when opening a form.

A form is typically opened with a set of data for the user to browse. This data set is controlled by the data set relationships and the where clause specified in the form record `Query` property. In some instances, you may want to open the same form with a limited set of data. To do so, in the BA diagram, add a filter on the relation to the form.

The `Filter` properties consist of the `Destination Field` to filter on (for example, `country.id`) and the `Source Field` values to use for filtering (for example, the value in `account.country_id`). When the form is opened, it will filter the results displayed based on the filter.

Field activation

An active field is one that can be input by the user. Field activation occurs when the application is in an edit mode (**MODIFY** or **ADD** modes).

CRUD operations (create, read, update, delete) apply to fields belonging to the business record master table thus only the business record master table fields can be edited and saved.

These fields can be edited directly by user input or automatically by ascending lookups.

In edit mode, active fields are:

- fields belonging to the business record master table
- fields running ascending lookups

Automatically deactivated fields:

- Formonly fields
- Unique key fields when the form is in MODIFY mode
- Fields defined as a foreignField in a business record master-detail relationship

This default behavior can be bypassed by adding code in the predefined BLOCK/POINTS. Field activation sections are centralized to ease customization.

Note: When the user is in edit mode (property `canModify` is true) and no editable field is present, the program stops with an FGL error. In addition, if all fields of the business record master are primary keys, they can not be modified even if `canModify` is true. In the case of a master-detail relationship, a `DISPLAY ARRAY / INPUT ARRAY` may be generated instead of an `INPUT ARRAY / INPUT ARRAY`.

Define queries and data order

Joins between tables referenced in the form are set up in the query property of the business record.

The **Edit query** dialog allows you to specify the joins between the tables in a record used for a form, report, or service.

Define a dynamically populated ComboBox

Figure 171: Combobox with Data

Populated Combobox Example

In this example, when the user is in an input mode and selects the ComboBox field, a list of the valid country descriptions are displayed. When the user selects a country description (`country.codedesc`) and accepts the selection, the `account` table is updated with the correct country information (`account.country`).

1. Modify the form. Find the master field that is to be replaced with a ComboBox of populated values and move it to another location on the form. You can convert this field to a Phantom widget as it will be needed for the code, but not needed to display to the user (`account.country` in this example).
2. On the Records tab, add the reference field to the master field in the same record. Right-click on the record and choose **Add Field**. (The `country.codedesc` field, the field representing the list of country descriptions, is added to the account record in this example.)
3. Right-click on the record and select **Edit Query**. Establish the join between the master table and reference table. (`account.country` is joined to `country.code` in this example.)
4. Select the reference field in the record and set its `lookup` property. This name will be used as a function name in the generated code that is triggered to perform the ascending lookup.

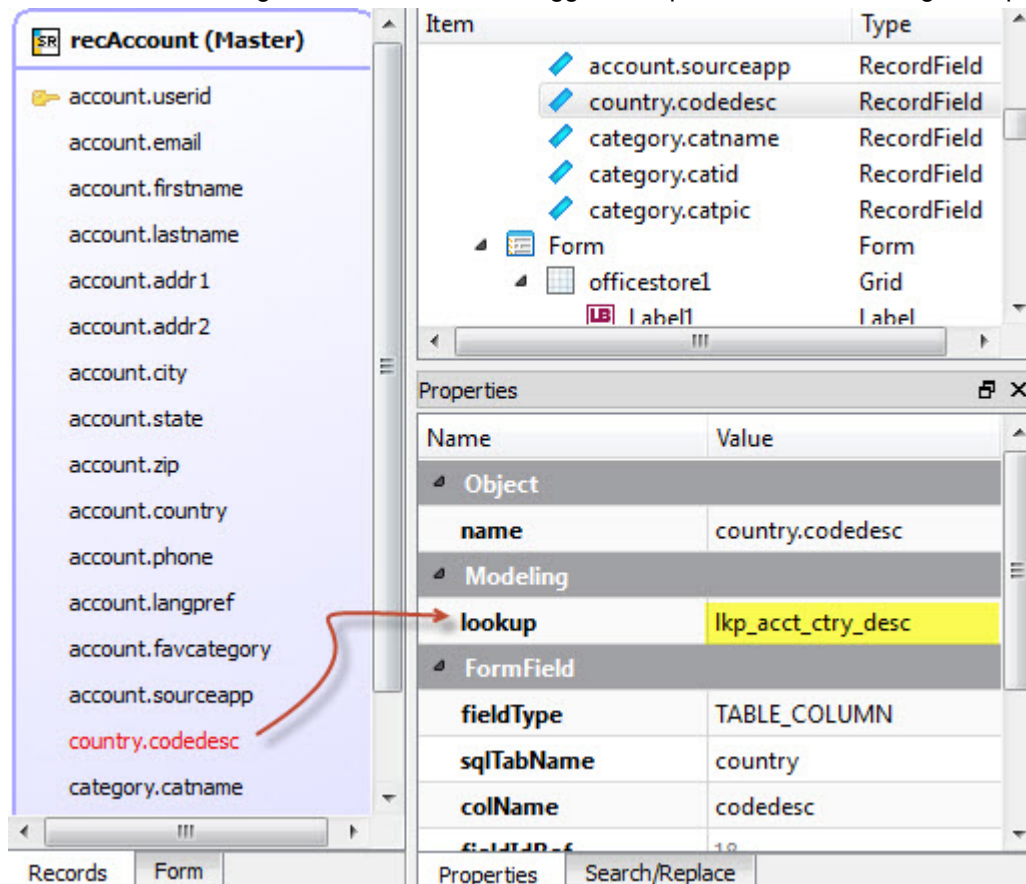


Figure 172: Lookup

5. Return to the **Form** tab. Notice that the new field (`country.codedesc`) has been added to the form in the upper left. Move it to its correct location and convert it to a ComboBox widget.
6. Set the `initializer` property on the ComboBox field to a unique name. This name will be used as a function name for the generated code that populates the ComboBox when it is built in the user interface.

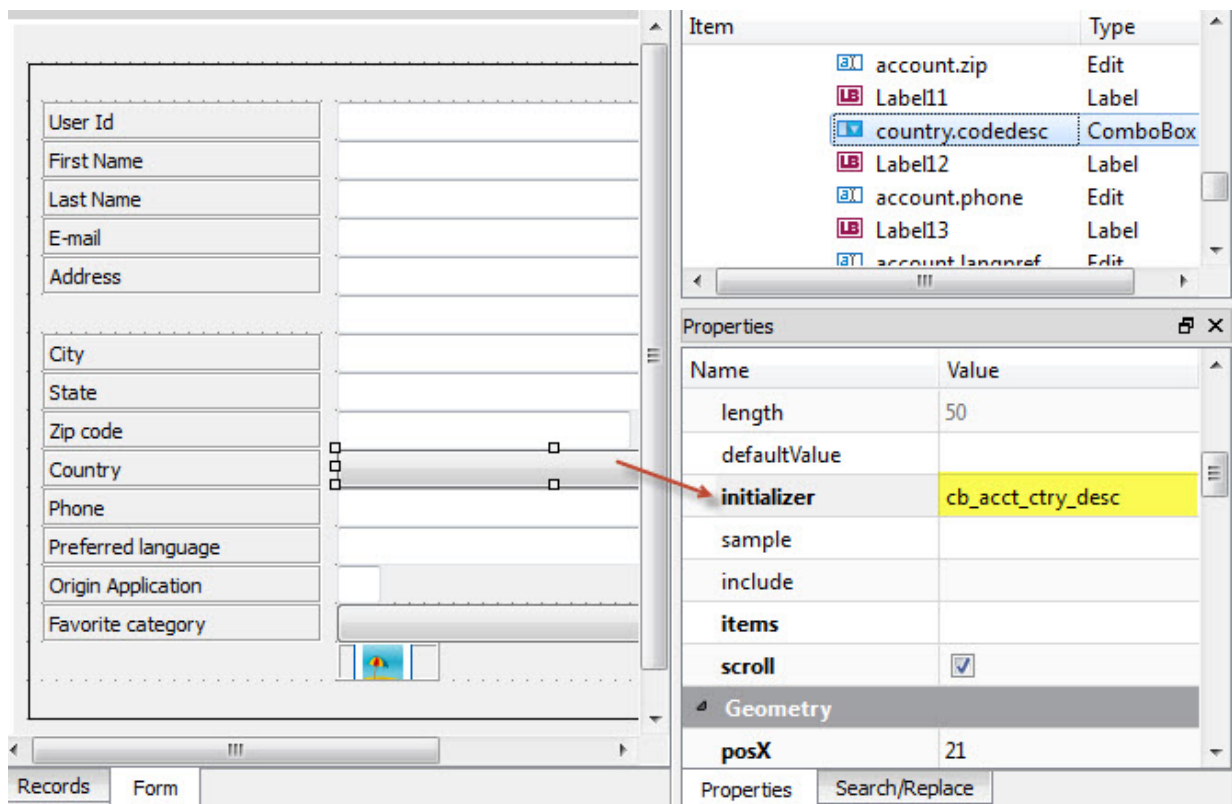


Figure 173: Combobox Properties

7. Save the form. **Build** and **Execute** the program.

Lookup fields

A lookup field is a field linked to another one inside a record. Lookup fields contain a value that is retrieved from a reference table instead of being input by a user.

There must be a join relationship between the master and reference tables inside a record. There are two types of lookup fields: descending and ascending.

descending

Descending lookup fields are automatically managed. A descending lookup field specifies that when a master field is updated, the related non-master fields will also be updated. The fields that are automatically updated are determined from the joins in the Query Editor. For this reason, the only type of lookup that is set up in the form is ascending.

ascending

Ascending lookup fields specify that when a field of a non master table is updated, the related master table field will be updated. An ascending lookup is implemented by entering a value in the `lookup` property on the field (that has sense only for fields of a non master table in the record) in the form record.

Composite fields in a lookup

If the lookup requires more than one field, set the `lookup` property with the same value on all these fields. The lookup is resolved with the set of values of these fields.

Descending lookup example

In this example, the form is primarily for the **account** table (master), but when a customer's account record is displayed or input, it also displays the corresponding country name from the **country** table (descending lookup).

The screenshot shows a software form titled 'Form1' with a menu bar (File, Edit, Navigate) and a toolbar with icons for Accept, Cancel, New, Modify, Delete, Search, and navigation (First, Previous, Next, Last). The form contains the following fields:

User Id	dupont	E-mail	jean@dupont.com
First Name	Jean	Last Name	Dupont
Status	OK	Address	3, rue d'Aboukir
Address		City	Paris
State		Zip code	75002
Country	FRA France		

The 'Country' field is highlighted with a red circle, showing 'FRA' and 'France'. The status bar at the bottom indicates '1/5' and 'OVR'.

Figure 174: Form showing lookup field

Define an ascending lookup field

Ascending lookup fields specify that when a field of a non master table is updated, the related master table field will be updated.

1. Open the form to be modified. A record of the form must include fields from the master table as well as the lookup field(s) from the reference (non master) table.
2. Confirm the join between the tables. Right-click on the record and select **Edit Query** to confirm the join between the master and reference (non master) table.
3. Select the Record tab and then the lookup field column to see its properties. Set the `lookup` property to a unique name (*lkp_acct_etry_desc* in this example). To implement a composite lookup, set the lookup property of both lookup fields to the same value.

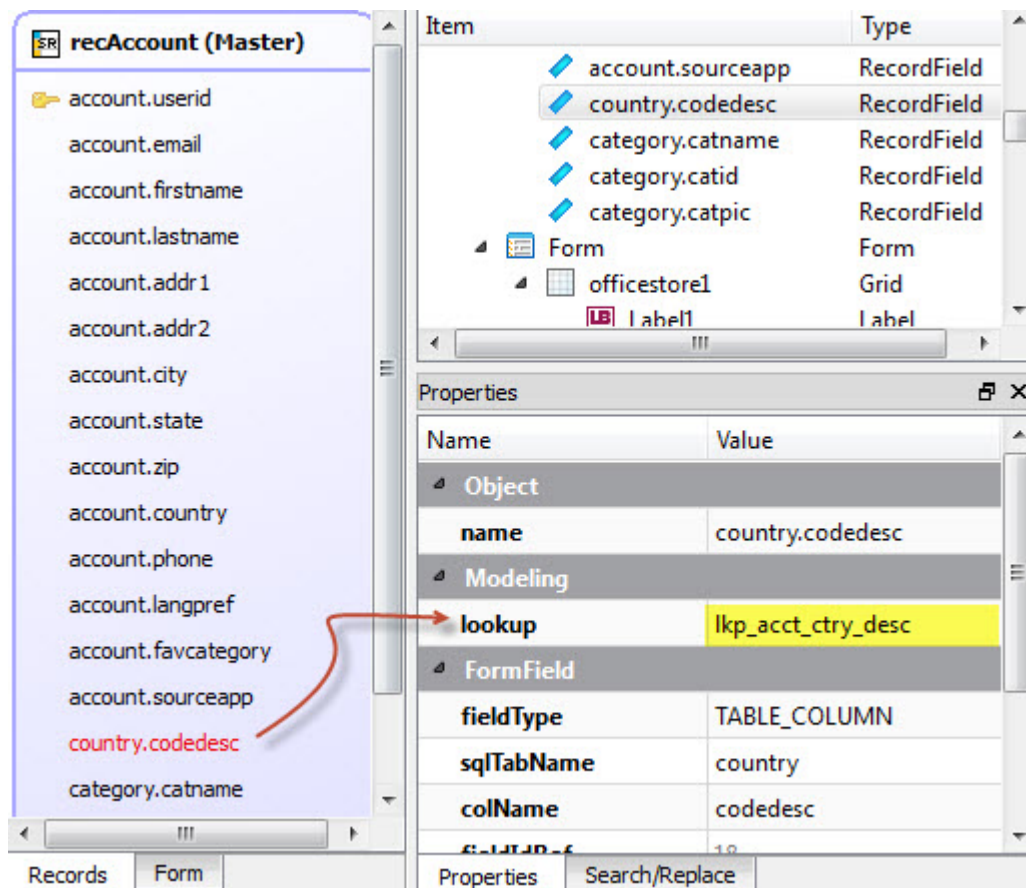


Figure 175: lookup Property

The generator will use this name in the function call that triggers the ascending lookup when the user modifies the reference (non master) field data. If the value is found in the reference table, it is used to update the master field. If it is not found in the reference table, the original values on the form remain intact.

4. Save the form. **Build** and **Execute** the program.

Add buttons to form

If you add a widget to your form that can trigger an action, such as a Button, you can associate an action from the generated program to the widget.

1. Open the form.
2. Select **Widget>>Button** from the menu.
3. Draw out a **Button** in a container on the form.
4. Provide the name of the action in the **name** property for the Button.

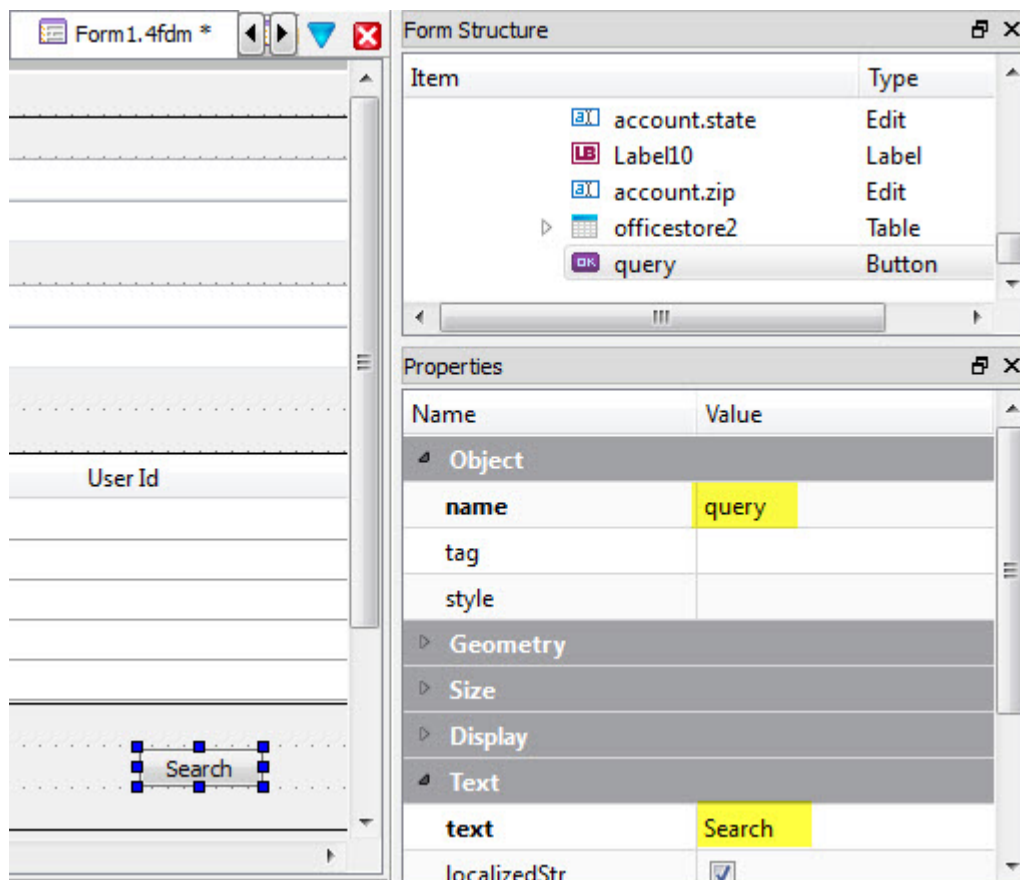


Figure 176: Add Button

Add formonly (nondatabase) fields to a form

Formonly fields are added by creating a non database field and making the record inactive. You must supply the business logic for the formonly field - it is not generated.

Master-detail forms

A Genero application can display a form that contains a master-detail relationship between two tables.

The user can search for a row in the master table, and the corresponding rows in the detail table will also be displayed. The values in rows from both tables can be added, deleted, or modified.

The form must contain fields from both tables and the table relationships must be set.

Adding custom code

This section includes topics on how to add custom code to the generated code.

- [Understanding what gets generated](#) on page 250
- [Finding the right place to customize](#) on page 259
- [Using POINTs and BLOCKs](#) on page 262

Understanding what gets generated

4gl files are generated for each diagram entity.

When a generated program is built, the generated 4gl files are compiled and linked with the other program files to create the executable application. The compiled binary versions are stored in the project's /bin directory.

The generated 4gl and XML files for each diagram entity are shown in the **Intermediate Files** folders in the project. This figure shows two generated files for the **Program** entity, `Account.4prg`.

Note: The XML file listed in the **Intermediate Files** folder are temporary files that consolidate all data from diagrams and are used to generate the 4gl code. They are not places to customize code, but can be useful to a template developer as these XML files are used as input to the code that is generated. See [How code is generated](#) on page 194.

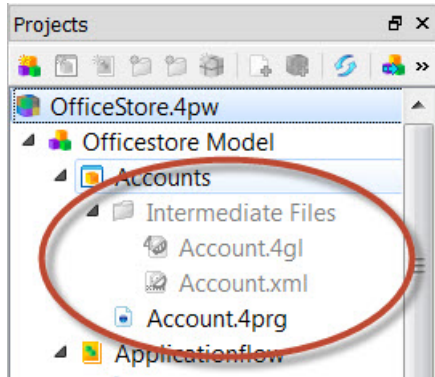


Figure 177: Intermediate Files

The default template set is designed to generate organized and functional code for a data-driven business application. See [The default template features](#) on page 199.

Each generated 4gl file has many functions. Each function has specific places called **POINTS** and **BLOCKS** where you can add your own code. [Code Files and Code Link Properties](#) can be used to directly access generated files and specific POINTS.

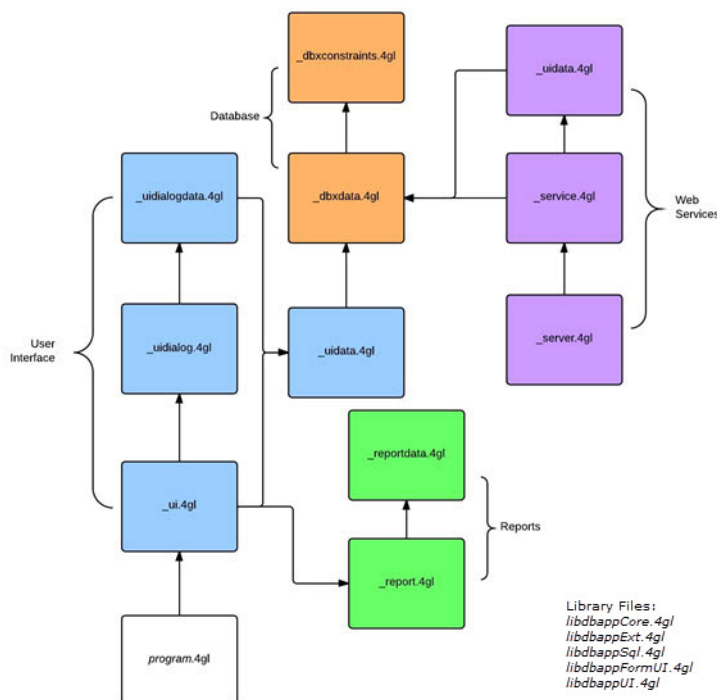


Figure 178: Generated files for the default template

Table 56: Generated files for the program (4prg)

File	Description
Program.4gl on page 254	This file contains the <code>MAIN</code> function. It defines global variables, connects to the data source with the <code>CONNECT TO</code> statement, loads styles and action defaults files, and performs some initialization tasks for the application.

Table 57: Generated files for the database (4dbx)

File	Description
Schema_dbxdata.4gl on page 258	This file manages the database <code>SELECT</code> , <code>INSERT</code> , <code>UPDATE</code> , and <code>DELETE</code> statements.
Schema_dbxconstraints.4gl on page 259	This file manages table and column constraints.

Table 58: Generated files for form entities (4fdm, 4fdz)

File	Description
Form_ui.4gl on page 255	This file contains functions called from <code>MAIN</code> . It opens the form based on the open mode, loads the Toolbar and Topmenu, launches functions to manage application states, calls <code>MENU</code> or <code>DIALOG</code> statements for the various states and calls functions in the <code>Form_uidialog.4gl</code> to retrieve and manage subdialogs.
Form_uidialog.4gl on page 256	This file defines the subdialogs and modular variables. It manages the states of actions, fields and subdialog and calls functions in the <code>Form_uidialogdata.4gl</code> .
Form_uidialogdata.4gl on page 257	This file defines the modular variables, manages UI data (clear, fetch, set default values), and calls functions in the <code>Form_uidata.4gl</code> .
Form_uidata.4gl on page 257	This file includes functions to retrieve and manage the database data. It defines the records for the database columns and tables, creates and fills a dynamic row of data, creates and fills a dynamic row of record keys, contains the SQL statements to <code>SELECT</code> , <code>UPDATE</code> , and <code>DELETE</code> rows in the database, and handles the SQL transactions.

Important: Reports and Web Services are not supported on mobile platforms.

Table 59: Generated files for the reports

File	Description
<code>Report_report.4gl</code>	This file contains the report driver for the report.
<code>Report_reportdata.4gl</code>	This file contains the SQL and BDL statements to fetch the report data.
<code>reportdata_report.rdd</code>	This file is a data file is generated for use when the developer creates the report definition file (4rp).

Table 60: Generated files for the web services

File	Description
<code>WebService_service.4gl</code>	This file handles business record type definition (only take public fields), defines the module variables (input and output variable for service's CRUD operations), and sets up the service and publishes the CRUD operations.
<code>WebService_uidata.4gl</code>	This file handles business record type definition, contains operations to insert / update / delete rows in the database, contains ascending lookup management.
<code>WebServiceServer_server.4gl</code>	This file contains the MAIN function, sets up the web service engine, contains the CONNECT TO BDL statement, registers services, and listens for incoming requests.
<code>WebService.xml</code>	The XML file contains the information about the web service needed to generate the logic in the 4gl files.
<code>WebServiceServer.xml</code>	Contains the information about the Webservice Server needed to generate the logic in the 4gl files.

Table 61: Library files used by a generated app

File	Description
<code>libdbappCore.4gl</code>	This file includes common functions for managing errors and strings.
<code>libdbappExt.4gl</code>	This file includes common functions for managing the interaction with mobile device peripherals. See Add mobile device features (Photo, Gallery, Phone, Mail, SMS, Contact, Maps, Barcode) on page 223
<code>libdbappFormUI.4gl</code>	This file includes common functions related to form transitions and states.
<code>libdbappSQL.4gl</code>	This file includes common functions related to SQL transactions, errors, and database statements.

File	Description
libdbappUI.4gl	This file includes common functions related to the user interface such as Topmenus, Toolbars, and front end type.

Program.4gl

This file contains the `MAIN` function. It defines global variables, connects to the data source with the `CONNECT TO` statement, loads styles and action defaults files, and performs some initialization tasks for the application.

The function(s) in this file include `POINT` and `BLOCK` sections where you can add your own code.

Table 62: The *Program.4gl* file

Generated Functions	Description
MAIN	The <code>MAIN</code> function of the program.

Form_ui.4gl

This file contains functions called from MAIN. It opens the form based on the open mode, loads the Toolbar and Topmenu, launches functions to manage application states, calls MENU or DIALOG statements for the various states and calls functions in the *Form_uidialog.4gl* to retrieve and manage subdialogs.

Table 63: The *Form_ui.4gl* file

Generated Functions	Description
uiOpenForm	Opens form according to the open mode.
uiOpenFormByKey	Opens form according to the open mode and position to the key.
uiAutomaton	The automaton function allowing the switch between modes (DISPLAY, MODIFY, ADD, SEARCH, EMPTY)
uiDisplay	Manages the DIALOG block for the DISPLAY mode.
uiInput	Manages the DIALOG block for the MODIFY and ADD modes.
uiConstruct	Manages the DIALOG block for the CONSTRUCT mode.
uiSearch	Manages the DIALOG block for the SEARCH mode.
uiEmpty	Manages the DIALOG block for the EMPTY mode.
initializeDefaultActions	Initializes the list of default actions.
initializeDefaultUISettings	Initializes the UI settings of the form.
action_	Launch a module.
validateCRUDOperationWrapper	Validate a CRUD operation for a record (INSERT, UPDATE, DELETE).
initialzeReports	Initializes the list of available reports.
processReport	Load, configure and run the current report.

Form_uidialog.4gl

This file defines the subdialogs and modular variables. It manages the states of actions, fields and subdialog and calls functions in the *Form_uidialogdata.4gl*.

Table 64: The *Form_uidialog.4gl*

Generated Functions	Description
<code>getDefaultField</code>	Get the default DIALOG field name.
<code>validateCRUDOperation</code>	Validate a CRUD operation (INSERT, UPDATE, DELETE).
<code>processCRUDOperation</code>	Process a CRUD operation (INSERT, UPDATE, DELETE). Check the concurrent access.
<code>synchronizeUI</code>	Synchronize the UI according an internal action returned by a CRUD operation.
<code>setAllCurrentRow</code>	Set the current row for each subdialog in the case of master-details to restore the context when the mode switches.
<code>uiDisplay</code>	DISPLAY ARRAY dialog.
<code>uiInput</code>	INPUT dialog.
<code>uiConstruct</code>	CONSTRUCT dialog.
<code>setActionStates</code>	Sets the state of the dialog actions to active or inactive according to the mode.
<code>setFieldActive</code>	Initialize fields state according to startup state.
<code>getRecordFieldList</code>	Gets the field list of a subdialog.
<code>navigate</code>	Navigation for current record.
<code>setAllCurrentRow</code>	Set the current row on all subdialogs whose container is a 'table' to restore the context.
<code>init</code>	Populate the referenced ComboBox.

Form_uidialogdata.4gl

This file defines the modular variables, manages UI data (clear, fetch, set default values), and calls functions in the *Form_uidata.4gl*.

Table 65: Form_uidialogdata.4gl

Generated function	Function features
seek	Gets the index of an item matching the given criteria.
fetchRowAndDetails	Fetch row and details data for record.
fetchAll	Fetch the data set for a record.
fetchRow	Fetch row data for record.
fetchDetails	Fetch details data for record.
clear	Clear record data.
clearRowAndDetails	Clear row and details data for record.
setDefaultValues	Initialize record with default values.
runUpdates	Run the treatments after the update of the field (like the descending, ascending lookups, ...). This function is especially called in the control blocks <code>ON CHANGE</code> , <code>ON ACTION zoom</code> .
runDescLookup	Generate descending lookup functions triggered by fields of the master table of the record. It calls 'DescLookup' defined in data then update the fields if lookup successes.

Form_uidata.4gl

This file includes functions to retrieve and manage the database data. It defines the records for the database columns and tables, creates and fills a dynamic row of data, creates and fills a dynamic row of

record keys, contains the SQL statements to `SELECT`, `UPDATE`, and `DELETE` rows in the database, and handles the SQL transactions.

Table 66: The *Form_uidata.4gl*

Generated Functions	Description
<code>getKey</code> s	Create and fill a dynamic array of Business Record(BR) Unique Keys (UK).
<code>getDataByKey</code>	Get data from a given Business Record (BR) Unique Key (UK) (may be a composite key).
<code>getDataArray</code>	Create and fill a dynamic array of Business Record (BR) fields.
<code>insertRow</code>	Insert a row in the database.
<code>updateRow</code>	Update a row in the database.
<code>deleteRow</code>	Delete a row in the database.
<code>deleteRowwithConcurrentAccess</code>	Delete a row in the database.
<code>checkRow</code>	Check a row in the database.
<code>checkRowConcurrentAccess</code>	Check a row in the database.
<code>DescLookup</code>	Retrieve data according the descending lookup after update of the field.
<code>AscLookup</code>	Retrieve data according the ascending lookup after update of the field(s).
<code>fillArray</code>	Populate an array of key/value pairs.
<code>computeFields</code>	Compute FORMONLY fields of the Business Record (BR).

Schema_dbxdata.4gl

This file manages the database `SELECT`, `INSERT`, `UPDATE`, and `DELETE` statements.

Find this file in the **Intermediate Files** folder listed with your database schema in your project.

Table 67: Schema_dbxdata.4gl

Generated function	Function features
<code>selectRowByKey</code>	Select a row identified by the primary key in the table.
<code>insertRowByKey</code>	Insert a row in the table and return the table keys.
<code>updateRowByKey</code>	Update a row identified by the primary key in the table.
<code>deleteRowByKey</code>	Delete a row identified by the primary key in the table.
<code>deleteRowByKeyWithConcurrentAccess</code>	Delete a row identified by the primary key in the table if the concurrent access is successful.
<code>deleteReferencingRowsByKey</code>	Delete rows referencing the primary key of the table.
<code>checkRowByKeyWithConcurrentAccess</code>	Check if a row identified by the primary key in the table has been modified or deleted.
<code>setDefaultValuesFromDBSchema</code>	Set data with the default values coming from the DB schema.

Schema_dbxconstraints.4gl

This file manages table and column constraints.

Find this file in the **Intermediate Files** folder listed with your database schema in your project.

Table 68: Schema_dbxconstraints.4gl

Generated function	Function features
<code>checkTableConstraints</code>	Check constraints on the table.
<code>checkUniqueConstraint</code>	Check the primary key uniqueness constraint on the table.
<code>checkColumnConstraints</code>	Check constraints on the column.
<code>checkFKConstraint</code>	Check the foreign key existence constraint on the table.

Finding the right place to customize

To determine where to add code to make your customization, use **Code Link** or **Code File** properties or consider these questions.

Direct access with Code Links

The most common places to customize the generated code are accessible from **Code Link** properties. **Code Link** properties provide direct access to the `POINT` in the appropriate generated file. For example,

the `Compute Fields` property on a **Form** record opens the correct generated 4gl file to the `POINT` location logically used for computing fields.

```
{<POINT Name="fct.recExample_computeFields.user">} {</POINT>}
```

You will find **Code Link** properties available on:

Meta-schema - Select the background of the Meta-schema diagram.

- **table**
 - Constraints
 - Key Uniqueness
 - Key Exists
 - Select
 - Insert (before)
 - Insert (after)
 - Update
 - Delete
 - Delete (Concurrent)
 - Cascade Delete
 - Optimistic Locking
 - Defaults
- **column**
 - Constraints (Column)

Form

- **Records tab**
 - **ManagedForm** - Select the background of the Form Records tab.
 - Open Window
 - UI Automaton
 - Display Mode Events
 - Input Mode Events
 - Search Mode Events
 - Empty Mode Events
 - Action State
 - **Record**
 - Select rows
 - Select row
 - Computed fields
 - Insert
 - Update
 - Delete
 - Delete (Concurrent)
 - Lookups (Descending)
 - Combobox initializer (Data)
 - Defaults (Dialog)
 - DISPLAY attributes
 - DISPLAY events
 - INPUT events
 - CONSTRUCT events
 - Action state

- Field Activation
- **Record field**
 - Field Activation
- **Form tab**
 - **Combobox widget**
 - Combobox initializer (Data)
 - Combobox initializer (UI)

Direct access with Code Files

Code Files properties are available for direct access to the appropriate generated code file.

You will find **Code Files** properties available on:

Meta-schema - Select the background of the Meta-schema diagram.

- Data opens [Schema_dbxdata.4gl](#) on page 258
- Constraints opens [Schema_dbxconstraints.4gl](#) on page 259

Form - Select the background of the Form Records tab.

- UI opens [Form_ui.4gl](#) on page 255
- Dialogs opens [Form_uidialog.4gl](#) on page 256
- Dialog Data opens [Form_uidialogdata.4gl](#) on page 257
- Data opens [Form_uidata.4gl](#) on page 257

Manually finding the right location

If a Code Link is not available to help guide you to a location for your custom code, these questions can help you determine where to customize.

Can the modification be done in a model?

For example, can you make a change to a property on an entity in the BA diagram to get the desired effect? This is the best first choice because the models drive the code generation. When you rebuild, the generated code reflects your changes. See [Modeling the application](#) on page 202.

Can the modification be done in an external resource file?

For example, you can modify the default Toolbar file that is loaded by the program. Modifying an external resource file does not affect the code generation if you use the default naming convention (dbapp.4tb, for example). When you rebuild, the modified resource file is loaded instead of the default one. See [Modify action defaults \(dbapp.4ad\)](#) on page 266, [Modify styles \(dbapp.4st\)](#) on page 267, [Modify the Topmenu \(dbapp.4tm\)](#) on page 267, [Modify the Toolbar \(dbapp.4tb\)](#) on page 267.

Is the modification database related?

Look at the functions in the [Schema_dbxdata.4gl](#) on page 258 and [Schema_dbxconstraints.4gl](#) on page 259 files. Look for a POINT in the function to place your custom logic. If a POINT is not available, modify the BLOCK of code. See [Using POINTs and BLOCKs](#) on page 262.

Is the modification for when the program begins or ends?	Look at the <code>MAIN</code> function in the Program.4gl on page 254 file. Look for a <code>POINT</code> in the function to place your custom logic. If a <code>POINT</code> is not available, modify the <code>BLOCK</code> of code. See Using POINTs and BLOCKs on page 262.
Is the modification related to the interaction with the user?	Look at the functions in the Form_ui.4gl on page 255, Form_uidialog.4gl on page 256, Form_uidialogdata.4gl on page 257, and Form_uidata.4gl on page 257. Look for a <code>POINT</code> in the function to place your custom logic. If a <code>POINT</code> is not available, modify the <code>BLOCK</code> of code. See Using POINTs and BLOCKs on page 262.
Is the modification related to reporting?	Look at the functions in the <code>_report.4gl</code> and <code>_reportdata.4gl</code> files. Look for a <code>POINT</code> in the function to place your custom logic. If a <code>POINT</code> is not available, modify the <code>BLOCK</code> of code. See Using POINTs and BLOCKs on page 262.
Is the modification related to web services?	Look at the functions in the <code>_service.4gl</code> and <code>_server.4gl</code> , and <code>_uidata.4gl</code> files. Look for a <code>POINT</code> in the function to place your custom logic. If a <code>POINT</code> is not available, modify the <code>BLOCK</code> of code. See Using POINTs and BLOCKs on page 262.
Is the modification repeating many times?	Consider modifying the template files. See BAM Template Developer Guide on page 930.

Using POINTs and BLOCKs

`POINT` and `BLOCK` sections are the areas in the generated code where you can add your own code.

Any code added in a `POINT` or `BLOCK` is preserved in the application *even when the application is rebuilt*.

Code added to a `POINT` or `BLOCK` section is preserved in a `.code` file that is used each time the application is compiled. If you are using Source Code Management, the `.code` file must be committed with the project. It is not necessary to commit the generated `4gl` files.

Tip: To remove all changes you have made, you can simply remove the `.code` and all generated `4gl` files.

POINT

`POINT` sections are located within each function `BLOCK`. The more granular `POINT` sections are located in all relevant locations for adding business logic such as in all control blocks (`BEFORE ROW`, `AFTER INPUT`, ...). Common uses of a `POINT` include defining your own variables, setting conditions on `SELECT` statements, adding or modifying actions in control blocks of interactive dialogs such as `CONSTRUCT`, `INPUT`, `MENU`, and changing the program flow.

Note: You are responsible for the validity of the code in a `POINT`.

Table 69: POINT examples

Description	POINT name	Example
Import an additional module(s).	<code>import</code>	<pre>{<POINT Name="import " Status="MODIFIED">} IMPORT FGL mylibrary</pre>

Description	POINT name	Example
		<pre>{</POINT>}</pre>
Add comments in the code.	comment	<pre>{<POINT Name="user.comments" Status="MODIFIED">} --Additional information about this module. {</POINT>}</pre>
Define modular scope variables.	define	<pre>{<POINT Name="define" Status="MODIFIED">} DEFINE myvar STRING {</POINT>}</pre>
Define local scope variables.	<i>function.define</i>	<pre>{<POINT Name="fct.uiOpenForm.define" Status="MODIFIED">} DEFINE myvar STRING {</POINT>}</pre>
Add additional function(s) to the module.	user.functions	<pre>--Add user functions {<POINT Name="user.functions" Status="MODIFIED">} FUNCTION dispmsg() MESSAGE "Program ending" END FUNCTION {</POINT>}</pre>
Add additional actions for each of the interactive dialog statements (DISPLAY, DISPLAY ARRAY, CONSTRUCT, INPUT, INPUT ARRAY and MENU).	userControlBlock	<pre>{<POINT Name="fct.uiInput.dlg.userControlBlocks" Status="MODIFIED">} ON ACTION myaction CALL dispmsg() {</POINT>}</pre>

BLOCK

Each function in the generated code is nested within a BLOCK section.

Note: You can change the behavior of the generated function, however once new code has been added to a BLOCK, you take responsibility for the validity of all the code in that BLOCK.

Status attribute

During the first application generation, POINT and BLOCK sections will only contain the Name attribute.

```
{<POINT Name="fct.uiDisplay.dlg.userControlBlocks">}{</POINT>}
```

When the code is changed, the POINT and BLOCK will include a Status attribute set to MODIFIED.

```
{<POINT Name="fct.uiDisplay.dlg.userControlBlocks" Status="MODIFIED">}
  ON ACTION test
  MESSAGE "testing..." {
```

```
</POINT>}
```

Revert a change to a POINT or BLOCK

Custom changes made to the source code in a POINT or BLOCK can be reverted.

1. Open your source code in Code Editor.
2. Select the area you want to revert.
3. Right-click on the selected POINT or BLOCK heading and use the **Point/Block >> Revert Point/Block** menu option. This will add a new attribute `Action="REVERT"`.
4. Compile the application and the changes made in the source will be removed.

Lost POINT or BLOCK

If a POINT / BLOCK is present in your source code but is no longer defined in the template file, the POINT / BLOCK will be considered "LOST". When the application is rebuilt, the `Status` attribute for the missing POINT / BLOCK will be set to LOST.

Lost sections, with their complete content intact, are commented out and put at the end of the regenerated source file. For example:

```
{<BLOCK Name="myBlockName" Status="LOST">}
-- DEFINE i INT -- DEFINE s STRING
-- {<POINT Name="myPointName" Status="MODIFIED"> } LET s = "Hello World"
{</POINT>} -- LET i = 10 -- {<POINT Name="myPointName2">} {</POINT>}
{</BLOCK>}
```

Note: The LOST status is not set on the POINT nested within the lost BLOCK, however the entire lost BLOCK is commented out including all nested POINT / BLOCK sections.

Modifying the look and feel

This section includes topics on how to modify the look and feel of the application.

- [Modify action defaults \(dbapp.4ad\)](#) on page 266
- [Modify styles \(dbapp.4st\)](#) on page 267
- [Modify the Topmenu \(dbapp.4tm\)](#) on page 267
- [Modify the Toolbar \(dbapp.4tb\)](#) on page 267

Default actions

Generated applications have default actions, which are triggered when the user clicks on an action view on the form (such as a Toolbar icon). The actions are enabled appropriately as the BDL interactive statements in the generated application are executed.

Default actions

The user interface for your BAM program is based on the form created for the program. When you set the **Functionality** properties on form records, you are specifying which default actions should be available to the user. For example, if the `canSearch` property is checked, the default actions needed to input criteria and search the database would be generated.

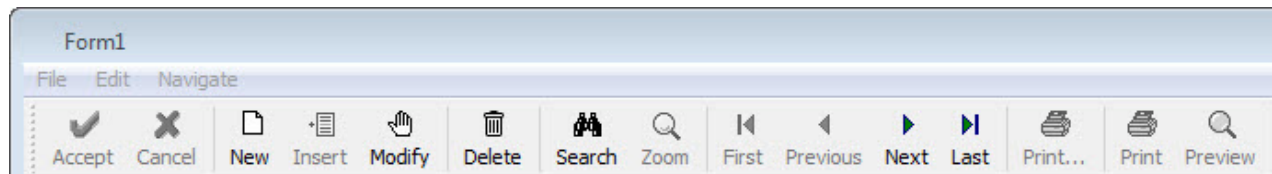
Actions can be programmatically enabled and disabled, hidden and shown, with methods such as `ui.Dialog.setActionActive()` and `ui.Dialog.setActionHidden()`. The text, image and other attributes of the action can be controlled with an action default file (4ad). See the *Genero Business Development Language User Guide*.

Table 70: Default Actions

Action Text	Action Name	Description
New	new	Adds a new record in the <i>master</i> , whether or not the focus is in the master or a detail.
Insert	insert	Appends a new record at the end of the list of the currently selected table - either master or detail.
Append	append	Adds a new record at the location of the current selection - either in a master or a detail. The append and new actions are equivalent if the focus is in the master.
Modify	modify	Update a record.
Search	query	Search the database table; enter criteria in the relevant fields and click Accept; or click Accept on an empty form to retrieve all the records in the database table.
Delete	delete	Delete a database record.
Zoom	zoom	Activate the zoom form.
First	firstrow	Navigate to the first record.
Last	lastrow	Navigate to the last record.
Next	nextrow	Navigate to the next record.
Previous	prevrow	Navigate to the previous record.

About the new, append, and insert actions

The `new`, `append`, and `insert` actions always create a new record. The `new` action only creates a master record. The `append` and `insert` actions create either a master record or a detail record. The `new` action is a global action. The `append` and `insert` actions are contextual according to the container. The `new` and `append` actions will always appear in the Toolbar/Topmenu. The `insert` action will appear in the Toolbar/Topmenu if there is at least one list on the CRUD form.



See the *Action rendering* topic in the *Genero Mobile Developer Guide* for information on the default action rendering in Android and iOS mobile apps.

Figure 179: Actions displayed in a desktop vs. mobile app

Report actions

Important: This feature is not supported on mobile platforms.

If a Report entity is implemented in the BA diagram, additional actions are enabled for the user. Which actions appear is controlled by the Report Options properties of the Form entity.

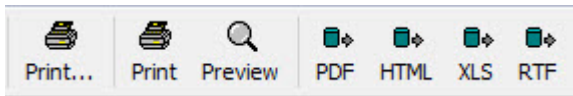


Figure 180: Default Toolbar with Report options

Table 71: Additional Actions for Reports

Action Text	Action Name	Description
Print	reportprint	Print a defined report.
Preview	reportpreview	Preview a defined report.
Print...	reportsetup	Select report and printer or export settings in Report Print Settings window. If no report design template (4rp) is available for the report, the Select Fields button is enabled to select fields for the report.
PDF	reportexportpdf	Export report to PDF format.
HTML	reportexporthtml	Export report to HTML format.
XLS	reportexportxls	Export report to XLS (Excel) format.
RTF	reportexportrtf	Export report to RTF (MS-Word) format.

Modify action defaults (dbapp.4ad)

You can modify the Action Default file used with the generated program.

The text, image and other attributes of the action can be controlled with an action default file (4ad). See the *Genero Business Development Language User Guide*.

The recommended procedure to modify the action default file used with the generated program is to create a new Action Default File (4ad) based on the dbapp.4ad template.

1. Select **File >> New >> Design >> Action Defaults (4ad)**.
2. Modify the 4ad file by adding, modifying, or deleting actions and action attributes.
3. Save the file to your project with the name dbapp.4ad.

Note: If you save the file to your project with the name dbapp.4ad, Genero Studio will use this file at runtime instead of the template. If you choose a different name, you will need to find the BLOCK in the generated code that calls the `ui.interface.loadActionDefaults()` method. Change the parameter name of the file name to your new file name.

Default Topmenu and Toolbar

The form for your generated program contains default action views (in a Topmenu and Toolbar) allowing the user to trigger the program actions.

Desktop applications

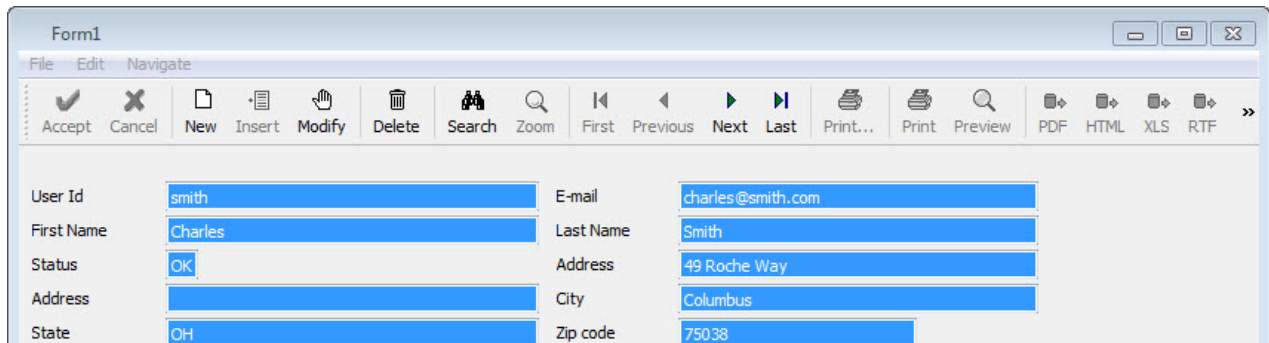


Figure 181: Application generated by the default template

Mobile apps

See the *Toolbar* and *Topmenu* topics in the *Genero Mobile User Guide* for information on the default action rendering in Android and iOS mobile apps.

Modify the Topmenu (dbapp.4tm)

You can modify the Topmenu used with the generated program.

The recommended procedure is to create a new Topmenu based on the `dbapp.4tm` template.

1. Select **File >> New >> Design >> Top Menu (4tm)**.
2. Modify the `4tm` file by adding, modifying, or deleting actions.
3. Save the file to your project with the name `dbapp.4tm`.

Note: If you save the file to your project with the name `dbapp.4tm`, Genero Studio will use this file at runtime instead of the template. If you choose a different name, you will need to modify the generated code to load your `4tm` file using the `ui.interface.loadTopMenu()` method.

Modify the Toolbar (dbapp.4tb)

You can modify the Toolbar used with the generated program.

The recommended procedure is to create a new Toolbar based on the `dbapp.4tb` template.

1. Select **File >> New >> Design >> Toolbar (4tb)**.
2. Modify the `4tb` file by adding, modifying, or deleting actions.
3. Save the file to your project with the name `dbapp.4tb`.

Note: If you save the file to your project with the name `dbapp.4tb`, Genero Studio will use this file at runtime instead of the template. If you choose a different name, you will need to modify the generated code to load your `4tb` file using the `ui.interface.loadToolBar()` method.

Modify styles (dbapp.4st)

You can modify the Style file used with the generated program.

The recommended procedure is to create a new Action Default File (`4ad`) based on the `dbapp.4ad` template.

1. Select **File >> New >> Design >> Style (4st)**.

2. Modify the `4st` file by adding, modifying, or deleting style and style attributes.
3. Save the file to your project with the name `dbapp.4st`.

Note: If you save the file to your project with the name `dbapp.4st`, Genero Studio will use this file at runtime instead of the template. If you choose a different name, you will need to find the `BLOCK` in the generated code that calls the `ui.interface.loadStyles()` method. Change the parameter name of the file name to your new file name.

BAM Reference

- [GSTSETUPDIR](#) on page 144
- [\\$\(generate\)](#) on page 268
- [tclsh](#) on page 269
- [\\$\(tcl\) - deprecated](#) on page 270
- [\\$\(blockpoint\)](#) on page 270
- [Business Application Modeling error messages](#) on page 271
- [Business Records error messages](#) on page 280
- [Business Application Diagram error messages](#) on page 285

BAM-specific environment variables

A subset of environment variables configure your Business Application Modeler (BAM) environment.

DBAPP_MOBILE

Defines whether the generated application is for a mobile device.

Set to `1` (TRUE) if the generated application is for a mobile device.

The `DBAPP_MOBILE` environment variable is used at the application generation level, to perform checks for unsupported functionality on mobile devices. The setting of this environment variable does not influence the generated code. The generated code will be the same regardless of the front end and regardless of the setting of the `DBAPP_MOBILE` environment variable.

For example, if the generated code includes an `INPUT ARRAY` statement, setting `DBAPP_MOBILE` results in warning [GS-13145](#); otherwise there is no warning. Likewise, if the generated code includes a `TreeView` or `ScrollGrid` container, warning [GS-13149](#) will be raised when `DBAPP_MOBILE` is set. The application, however, is still created, and in both cases the program will fail at runtime as the feature or container is not supported on mobile devices.

GSTSETUPDIR

Defines the BAM application generator template directory. Changing this variable launches synchronization from the server and reloads the templates.

Select the default environment set or create a new one that includes the `GSTSETUPDIR` specifying the location of the template directory to be used.

\$(generate)

The `$(generate)` command creates an intermediary XML file from modeled entities.

Syntax

```
$(generate) [options]
```

1. *options* are described in [Table 72: \\$\(generate\) options](#) on page 269.

Options

Table 72: \$(generate) options

Option	Description
<code>-oldModelFormat</code>	Generate XML file with prior format.
<code>-depth <i>depthNumber</i></code> (deprecated)	This feature is deprecated. Use <code>depth</code> attribute on <code>Item</code> elements in <code>settings.agconf</code> instead. For example, in a BA diagram specify the number of relations to traverse while generating the model. Outgoing relations are traversed. The model always contains the incoming and outgoing relation for the current item, but if the depth limit is reached, the target item definition is not generated. The depth is an integer starting from 0, or <code>unlimited</code> to traverse the complete application.
<code>-ba <i>baFilePath</i></code>	Path to the BA diagram (<code>4ba</code>). The path of the <code>4ba</code> file in the current project workspace is stored in the <code>\$(BAFilePath)</code> project manager variable.
<code>-o <i>outputFilePathPath filename</i></code>	Path of the generated XML file. <code>filename</code> is the generated XML file name. Uses the filename, replacing the extension with <code>xml</code> .

Usage

The `$(generate)` command is used in build rules for generated programs. [Predefined node variables](#) on page 366 can be used in the command.

```
$(generate) -ba "$(BAFilePath)" "$(InputPath)"
```

tclsh

The `tclsh` executable generates the final file by using both a Tcl template file and the intermediary XML file created by the `$(generate)` command.

The executable is `GSTDIR/bin/tclsh` (for UNIX™) and `GSTDIR/bin/tclsh.exe` (for Windows™).

Syntax

```
tclsh script.tcl [arguments]
```

1. Name of the intermediary XML file created by the `$(generate)` on page 268 command for the current item.

Usage

`tclsh` is used in build rules for generated programs. [Predefined node variables](#) on page 366 can be used in the command.

```
tclsh "$(TemplateDir)/tpl/genprg.tcl" "$(InputDir)/$(InputBaseName).xml"
```

\$(tcl) - deprecated

\$(tcl) is deprecated, tclsh is used instead of \$(tcl) to generate files. The \$(tcl) command launches the TCL interpreter and generates the final file by using both a Tcl template file and the intermediary XML file created by the \$(generate) command.

Syntax

```
$(tcl) [options] xml_filename
```

1. *options* are described in [Table 73: \\$\(tcl\) options](#) on page 270.
2. Name of the intermediary XML file created by the [\\$\(generate\)](#) on page 268 command for the current item.

Options

Table 73: \$(tcl) options

Option	Description
-tpl <i>templatefilename</i>	Tcl template file name.
-o	Path of the file to which to write output.

Alternative syntax

Syntax similar to the command line Tcl Interpreter is available from version 2.41.

```
$(tcl) templatename templatearguments
```

Usage

The \$(tcl) is used in build rules for generated programs. [Predefined node variables](#) on page 366 can be used in the command.

```
$(tcl) "$(TemplateDir)/tpl/main.tcl" "$(InputDir)/$(InputBaseName).xml"
```

\$(blockpoint)

The \$(blockpoint) command manages user added code by extracting or injecting code between BLOCK and POINT tags in a generated 4gl file.

Syntax

```
$(blockpoint) [options] "filename(s)"
```

1. *options* are described in [Table 74: \\$\(blockpoint\) options](#) on page 270.
2. *filename* is the generated 4gl file(s) separated by a space.

Options

Table 74: \$(blockpoint) options

Option	Description
-extract	Extract diff between <i>filename</i> and generated part of <i>filename.codefile</i> .

Option	Description
-storeGenerated	Store the generated part of <i>filename.code</i> with the content of <i>filename</i> .
-inject	Inject the diff part of <i>filename.code</i> in the <i>filename</i> .
-commentStart	Comment start pattern.
-commentStart2	Line comment pattern.
-commentEnd	Comment end pattern.
-code [.code file path]	Specifies the name of the .code file. If there is only one generated source file (4gl), the .code file uses (by default) the same name as the source file, otherwise -code is mandatory.

Usage

The `$(blockpoint)` command is used in build rules for generated programs. [Predefined node variables](#) on page 366 can be used in the command.

```
$(blockpoint) -code "$(InputDir)/$(InputBaseName).code"
               -extract "$(InputDir)/$(InputBaseName).4gl"
```

```
$(blockpoint) -storeGenerated -code "$(InputDir)/$(InputBaseName).code"
               -inject "$(InputDir)/$(InputBaseName).4gl"
```

Business Application Modeling error messages

A list of BAM error messages. For messages that are not self-explanatory, additional information is provided.

Table 75: Business Application Modeling Error Messages

Number	Description
GS-13001	Cannot load file. The file cannot be loaded; depending on the error, the message can change: <ul style="list-style-type: none"> Unknown node: an unknown node is present in the settings file Cannot register node: a node cannot be added to the file format (it's invalid or already present) Empty extension: a required file extension is empty. Check file path, format and permissions.
GS-13002	Cannot save file. Check file path, format and permissions.
GS-13003	Template not found. Check template directory path.
GS-13004	Unknown item %1. The <code>setting.agconf</code> template file format is incorrect. Validate it using the XML schema (or open it in Code Editor) and fix the errors.

Number	Description
GS-13006	settings.agconf version %1 is not supported, use version %2. Modify setting.agconf to match version 2 XML schema.
GS-13007	%1 BLOCK(S) or POINT(S) end tag is(are) missing. Fix the BLOCK / POINT..
GS-13008	Incorrect end BLOCK or POINT tag type. Fix the BLOCK / POINT.
GS-13009	End BLOCK or POINT tag does not correspond to an open tag. Fix the BLOCK / POINT.
GS-13010	%1 with name %2 is already defined. Rename the BLOCK / POINT.
GS-13011	A BLOCK or POINT cannot be a child of a POINT tag. Rename the BLOCK / POINT.
GS-13012	Renamed %1: %2 to %3. Preprocessor message that a BLOCK / POINT has been renamed: %1 = BLOCK or POINT %2 = old BLOCK / POINT name %3 = new BLOCK / POINT name
GS-13013	Lost %1: %2. %1 = BLOCK or POINT %2 = BLOCK / POINT name
GS-13014	Adding entries outside BLOCK or POINT not allowed. Add a BLOCK / POINT in the templates or remove your code.
GS-13015	\$(agcomp) is deprecated, please prefer using \$(generate), tclsh and \$(blockpoint) commands. Modify the build rule to use the new commands.
GS-13016	Modified %1: %2. %1 = BLOCK or POINT %2 = BLOCK / POINT name
GS-13017	Unknown property. An unknown property has been set in the settings file. Remove the property or change its name.
GS-13019	Cannot open file : extraction load failed. Check the rights on the generated source file.

Number	Description
GS-13020	<p>Cannot open file : injection save failed.</p> <p>Check the rights on the generated source file.</p>
GS-13021	<p>Cannot open file : update load failed.</p> <p>Check the rights on the generated source file.</p>
GS-13022	<p>Cannot open file : update save failed.</p> <p>Check the rights on the code file.</p>
GS-13023	<p>Code file load failed.</p> <p>Check the rights on the code file or check that the code file contains valid XML content.</p>
GS-13024	<p>Unknown argument %1 to Application Generator Block & Point task.</p> <p>The command \$(blockpoint) contains an unknown argument in the build rule. Check the argument %1 of \$(blockpoint) command in the build rule.</p>
GS-13025	<p>Missing -depth <i>int</i> argument to Application Generator compilation task.</p> <p>The Application Generator build rule requires the -depth argument in the \$generate command. Add the -depth argument to the \$(generate) command in Application Generator build rules.</p>
GS-13026	<p>Invalid -depth argument for Application Generator compilation task, required positive integer or 'unlimited'.The \$(generate) command requires a positive number (or keyword unlimited). Fix the argument in the build rule command.</p>
GS-13027	<p>Unknown argument %1 to Application Generator compilation task.</p> <p>Command or tclsh or \$(agcomp) contains an unknown argument in the build rule. Check the argument %1 of \$(generate) or tclsh or \$(agcomp) command in the build rules.</p>
GS-13028	<p>Invalid -endComment <i>string</i> argument, it cannot be used without -startComment <i>string</i> argument.</p> <p>The command \$(blockpoint) contains an -endComment argument without -startComment argument in the build rules. Remove the -endComment argument or add a -startComment argument in the build rule.</p>
GS-13029	<p>Error decoding %1 using codec %2 (encoding=%3)</p> <p>The file contents cannot be decoded with the codec, the encoding specified in the environment does not correspond to the file and it cannot be read. Change the encoding so that it supports all the file characters.</p>
GS-13030	<p>Error encoding %1 using codec %2 (encoding=%3)</p> <p>The file contents cannot be encoded with the codec, the encoding specified in the environment does not correspond to the file and it cannot be written to. Change the encoding so that it supports all the file characters.</p>

Number	Description
GS-13031	<p>Missing codec for encoding %1</p> <p>There is no default text codec for the specified encoding. Use another alias for this encoding, update encodingMap.xml, or add a new POSIX charmap.</p>
GS-13032	<p>Malformed BLOCK/POINT start tag</p> <p>The BLOCK/POINT start tag has a wrong syntax. Restore the right syntax.</p>
GS-13033	<p>Unexpected characters outside text blocks : %1</p> <p>Some characters have changed outside the topmost block, which is not supported. Insert a new toplevel block around the code you want to modify and regenerate the code.</p>
GS-13034	<p>Missing argument -code in the \$(BlockPoint) command</p> <p>The \$(BlockPoint) command is used with multiple generated files, the -code argument is mandatory. Add the -code argument.</p>
GS-13035	<p>Missing files in the \$(BlockPoint) command</p> <p>The \$(BlockPoint) command is used without generated files arguments. Add the generated files list.</p>
GS-13036	<p>%1 contains code not managed in the \$(BlockPoint) command</p> <p>Some file managed by the .code were not updated by the \$(BlockPoint) command, the .code is not completely up to date. Add the missing files to the command line in the build rule or remove the old file from the .code if they are no longer useful.</p>
GS-13037	<p>.code file version %1 is not supported</p> <p>The current version of Genero Studio does not support this version of .code file. Upgrade Genero Studio.</p>
GS-13038	<p>.code file encoding %1 differs with current one</p> <p>The .code file encoding is not the same as the build encoding, the resulting files may be incorrect.</p> <p>Change the build encoding (the LANG variable for example) to match the .code file encoding or rewrite the .code file with the current encoding.</p>
GS-13039	<p>Resolve conflict before compiling</p> <p>The code file id does not match the source file id, thus a resulting conflict file. You must merge the differences manually. This may occur after updating from the SVN repository.</p> <p>Merge the .conflict file and the source file, then delete the .conflict file.</p>
GS-13040	<p>Cannot write conflict file</p> <p>The code file id does not match the source file id, thus a resulting conflict file which cannot be written to the disk.</p> <p>Check the directory permissions.</p>
GS-13042	<p>'-depth' argument of \$(generate) command is deprecated, use 'depth' attribute on 'Item' elements in settings.agconf.</p>

Number	Description
GS-13100	<p>The 'Open Mode' property is not set.</p> <p>The Open Mode property value is empty. The available values are: DISPLAY, MODIFY, ADD, SEARCH and EMPTY. Check settings.agconf and ensure that the initialValue attribute of the <DynamicProperty> node having name = "openMode" contains one of the available values.</p>
GS-13101	<p>The 'Action' property is mandatory.</p> <p>The Action property is mandatory when a relation is defined between modules (4fdm to 4fdm) or between module and zoom (4fdm to 4fdz) and the relation type=Relation. Check that the Type property of the relation is Relation. Check that the Action property of the relation is not empty.</p>
GS-13102	<p>No report file is defined. Default layout will be used.</p> <p>If the Report File property is empty, the default layout (ASCII mode) will be used when a relation is defined between a module and a report (4fdm to 4rd) and the relation type=ReportRelation. This is a warning message. Check that the Type property of the relation is ReportRelation. Check that the Report File property of the relation is not empty.</p>
GS-13103	<p>The 'Type' property of the relation is invalid.</p> <p>The type of relation is not valid between entities: module (4fdm), program (4prg), zoom (4fdz), report (4rd). Check the Type property of the supported relation and ensure that the relation is supported.</p>
GS-13104	<p>The relation is not supported.</p> <p>Check the entity relations.</p>
GS-13105	<p>Any relation to a program is not supported.</p> <p>Check the entity relations.</p>
GS-13106	<p>The 'Report file' property contains an absolute path. Prefer a relative path.</p> <p>Absolute paths are not recommended because the project will not be portable. Use a relative path. Be sure that your Genero environment variables are correctly set to search for your resource files.</p>
GS-13107	<p>A unique key field can only be defined on the master table.</p> <p>All fields making up the unique key of a record must be fields of the master table of the record. Uncheck the Unique Key property of fields which are not of the master table of the record.</p>
GS-13108	<p>Duplicate table name in the FROM clause.</p> <p>The joins between a pair of tables must have the same join operator. Check that there are not multiple join operators for the same pair of tables.</p>
GS-13109	<p>Field updated via ascending lookup is mandatory in the record.</p> <p>The field that is updated via an ascending lookup is mandatory in the master table of the record. Add the field to the master table of the record. Currently, the field is only defined in the Query clause of the record. If you don't want to see this field</p>

Number	Description
	on your form, use the phantom widget. Or, delete the lookup property of the field which triggers the ascending lookup.
GS-13110	<p>Multiple master records are not supported. Define a relation between records.</p> <p>When there are several records in a managed Form, only one can be the master. This error occurs when at least one record is not linked by a relation. In the Records view, check that all records are linked by a relation.</p>
GS-13111	<p>Invalid relation, <i>primaryField</i> and <i>foreignField</i> must have same number of fields.</p> <p>A relation must have the same number of primary fields and foreign fields. Modify the relation fields so that a primary field corresponds to each foreign field.</p>
GS-13112	<p>The 'zoom' action in the Toolbar/Topmenu won't be generated, an action is already named 'zoom'.</p> <p>Rename your action to a name other than zoom.</p>
GS-13113	<p>Several relations (\$1) use the same action name (\$2 - \$3).</p> <p>Several relations use the same combination of Action/Source Record property values.</p> <p>\$1 - Number of relations having the same combination of Action/Source Record.</p> <p>\$2 - Action name.</p> <p>\$3 - List of record names.</p> <p>Rename Action or change the Source Record to have a unique couple.</p>
GS-13114	SQL statement 'FULL OUTER JOIN' is vendor proprietary SQL syntax.
GS-13115	<p>Invalid filename. A filename must be a BDL identifier.</p> <p>Filename of 4prg, 4fdm, 4fdz, 4rd contains invalid characters. Identifiers must confirm to these rules:</p> <ul style="list-style-type: none"> • It must include at least one character, without any limitation in size. • Only ASCII letters, digits, and underscore (_) symbols are valid. • No blanks, hyphens, and other non-alphanumeric characters. • The initial character must be a letter or an underscore. • It is recommended to always write identifiers in lower case. <p>Rename the file to a valid file name.</p>
GS-13116	<p>Several comboboxes use the same initializer name.</p> <p>Make all initializer names unique.</p>
GS-13117	<p>A ComboBox is possibly not initialized.</p> <p>Set the initializer or the items property of the ComboBox.</p>
GS-13118	Table \$1 unused in query.

Number	Description
	A database table is used in the record, but there is no join for it in the query. Add a join for the table in the record query.
GS-13119	<p>Duplicated joins are forbidden in the query.</p> <p>Remove the duplicated joins.</p>
GS-13120	<p>Building \$1 generates an XML file (version=\$2) which cannot be handled by the templates (version=\$3).</p> <p>The build process of the file (\$1) has been aborted because the intermediate XML file has a version (\$2) which mismatches the template version (\$3).</p> <p>Use a valid template set defined in Application Generator Preferences. Verify build rules for the appropriate file type.</p>
GS-13121	<p>The \$1 property is not supported.</p> <p>In the Business Application diagram, the <code>openMode</code> or <code>defaultMode</code> property defined on a relation between a CRUD Form and a Zoom Form is not supported. Only <code>DISPLAY</code> and <code>SEARCH</code> values are currently supported on a Zoom. Change the <code>openMode</code> or <code>defaultMode</code> property to a supported value.</p>
GS-13122	<p>The \$1 property is missing in <code>settings.agconf</code>.</p> <p>The build process has been aborted because the \$1 <code>dynamicProperty</code> is missing in the <code>settings.agconf</code> configuration file.</p> <p>Check <code>settings.agconf</code> and confirm:</p> <ul style="list-style-type: none"> • The <code><DynamicProperty></code> node with <code>name="\$1"</code> is defined in the <code><BusinessApplication></code> section. • The <code>dynamicProperties</code> attribute of the item raising the error contains \$1.
GS-13123	<p>No CRUD function will be generated for the \$1 table because no primary key has been defined.</p> <p>A primary key needs to be defined for the given table in order to have its CRUD functions generated when compiling the database schema.</p>
GS-13124	<p>Record functionalities are incompatible with the \$1 property value (\$2).</p> <p>On a CRUD Form, if the <code>openMode</code> or <code>defaultMode</code> property is:</p> <ul style="list-style-type: none"> • <code>ADD</code>, the master record must have the functionality <code>canAdd</code> activated. • <code>EMPTY</code>, the master record must have the functionality <code>canEmpty</code> activated. • <code>DISPLAY</code>, all records must have the functionality <code>canDisplay</code> activated. • <code>MODIFY</code>, at least one record must have the functionality <code>canModify</code> activated. <p>On a CRUD Form or Zoom Form, if the <code>openMode</code> or <code>defaultMode</code> property is <code>SEARCH</code>, at least one record must have the functionality <code>canSearch</code> activated.</p> <p>On incoming relations of a CRUD Form or Zoom Form, the <code>openMode</code> and <code>defaultMode</code> property must follow the same rules as those forms.</p> <p>To fix the error, change the <code>openMode</code> or <code>defaultMode</code> property value or change the functionality of the record.</p>
GS-13125	\$1 doesn't exist or is not unique\$2, please have a look in the XML intermediate file.

Number	Description
	An XPath cannot be resolved. Check the XML intermediate file.
GS-13126	<p>The number of source fields defined on the relation doesn't match the number of unique key fields defined in the destination item.</p> <p>Change the number of source fields defined on the relation.</p>
GS-13127	<p>Destination fields defined on a relation to a zoom are unused in the code generation.</p> <p>The destination fields defined on a Zoom relation will not appear in the generated 4gl code.</p>
GS-13128	<p>The table 'seqreg' is missing in your database schema, therefore SERIAL fields cannot be handled by the templates.</p> <p>A database schema field of datatype SERIAL can only be handled by using the seqreg table. This table contains a list of table names and values, a value is the last SERIAL created for a given table name. In case the couple (table name, last value) does not exist, a new record will be created and the SERIAL value will start at 1.</p> <p>Create the table seqreg in your database schema file.</p>
GS-13129	<p>Multiple SERIAL fields for the table \$1 cannot be handled by the templates.</p> <p>Only one SERIAL field can be managed per database table. This is inherent to the seqreg table.</p> <p>Modify the table in you database schema file.</p>
GS-13130	<p>Business Application diagram is missing in the project.</p> <p>Add a Business Application Diagram to the project.</p>
GS-13131	<p>Unique key must be a database primary or secondary key of the master table.</p> <p>All fields making up the unique key of a record must be either a database primary key or secondary key of the master table (unique constraint).</p> <p>Change the unique key property of fields which are not a database primary key or secondary key of the master table or update the 4dbx schema.</p>
GS-13132	<p>Missing master table for \$1 record.</p> <p>Select a table in the master table property.</p>
GS-13133	<p>The field \$1 defined on the relation does not exist in Records view.</p> <p>In the Business Application diagram, when a relation is defined between items, the Source Field and the Destination Field must exist in the Records view.</p> <p>Check the Source Field and/or Destination Field property of the relation or open the appropriate item and add the missing fields.</p>
GS-13134	<p>Invalid relation, Source and Destination must have same number of fields.</p> <p>In the Business Application diagram, a relation between Forms must have the same number of fields in the Source Field and Destination Field.</p>

Number	Description
	Check the Source Field and/or Destination Field property of the relation.
GS-13136	<p>The Service Name property is mandatory.</p> <p>The property Service Name of a WebService item must have a value in the Business Application Diagram.</p> <p>Set the Service Name property value.</p>
GS-13137	<p>The Namespace property is mandatory. The property Namespace of a WebService Server item in a Business Application diagram must have a value.</p> <p>Set the Namespace property value.</p>
GS-13138	<p>Several web services use the same name.</p> <p>Any web service registered to the same Web Service Server must have a unique name.</p> <p>Ensure that the property Name of a web service is unique.</p>
GS-13139	<p>The \$1 action is a reserved action name.</p> <p>In the Business Application Diagram, when a relation is defined between items, the Action property can not be a reserved action name. The list of reserved action names is defined in the action defaults file (dbapp.4ad).</p> <p>Set the value of the Action property to a non-reserved action name.</p>
GS-13141	<p>\$1 - Check LANG variable or add entry in file encoding.tcl where \$1 is the value of the unsupported encoding</p> <p>The LANG environment variable is set, but the encoding part is either invalid (change the value of the LANG environment variable) or not present in the supported encoding array (add an entry in encoding.tcl).</p>
GS-13142	<p>All fields must be part of the same record.</p> <p>Choose fields from the same record.</p>
GS-13143	<p>The record \$1 defined on the relation does not exist in Records view.</p> <p>In the Business Application diagram, when a relation is defined between items, the Record Source must exist in the Records view. Check the Source Record property of the relation or open the source entity and add the missing record.</p>
GS-13144	<p>The 'Source Record' property is missing.</p> <p>The Source Record property is missing in the Business Application diagram. When a relation is defined between items, if the Row Bound property is checked on an outgoing relation, the Record Source must be defined if there is more than one record in the form. It is implicit if there is only one record.</p> <p>Check the Source Record property of the relation or uncheck the Row Bound property.</p>
GS-13145	<p>The \$1 functionality is not supported for \$2 container on mobile devices.</p> <p>Some functionality is not supported by the template. For example canAdd, canModify, and canSearch are not supported for tables.</p>

Number	Description
	Uncheck the functionality.
GS-13146	Several records \$1 are candidate for the 'Source Record' property. Remove the ambiguity and choose a record.
GS-13147	The 'Row Bound' property is not supported for GRID container. Uncheck the Row Bound property or choose the table container in the form.
GS-13148	The 'Source Field' property is missing. The position Source Field is required when the destination of the relation is a zoom.
GS-13149	The '\$1' container is not supported on mobile devices. Some containers such Tree and ScrollGrid are not supported on mobile devices. Change container to a supported container.
GS-13150	'Display, Add, Modify, Search are all disabled for record '\$1' This message is a warning, not an error. It simply tells you that all functionality is disabled for the record in question. Verify that having no functionality selected for the record is intentional.

Business Records error messages

A list of Business Records error messages. For messages that are not self-explanatory, additional information is provided.

Table 76: Business Records Error Messages

Number	Description
GS-24001	Error loading file <i>file</i> . An error occurred loading the file. Check the file name and permissions .
GS-24002	Malformed XML. XML file content is invalid. Select the correct file or correct the XML.
GS-24003	File <i>file</i> not found in Business Application diagram (<BA file name>). A referenced element is not found in the Business Application diagram. Select the correct file.
GS-24004	Invalid value for %1 property. Check the property syntax in the documentation and correct any errors.
GS-24005	Conflicted item. The Business Application diagram and Business Record unique id do not match.

Number	Description
	Open the 4ba file and resolve the conflict.
GS-24201	<p>Missing master Record.</p> <p>The document must contain at least one record that is master.</p> <p>Create a record.</p>
GS-24202	<p>Unique query key must be set.</p> <p>At least one field must be declared the unique key in a record.</p> <p>Set a unique key.</p>
GS-24203	<p>Record used in relation must be active.</p> <p>Change the record to active, or remove the relation.</p>
GS-24204	<p>Empty relation.</p> <p>This relation has no field definition.</p> <p>Add foreign/primary (or source/destination) fields.</p>
GS-24205	<p>Database table column referenced more than once in the form.</p> <p>In one form, a database table column can be referred only once. Table aliases should be used to attach more than one field to a database.</p> <ul style="list-style-type: none"> • Select one of the wrong Formfields and change the fieldType attribute from table_column to table_alias. • Select one of the wrong Formfields and change fieldType attribute to non_database.
GS-24206	<p>Missing master table for record record.</p> <p>Set the master table property.</p>
GS-24207	<p>Table %1 unused in query.</p> <p>A database table is used in the record, but there is no join for it in the query.</p> <p>Add a join for the table in the record query.</p>
GS-24208	<p>No schema attached.</p> <p>A database is required for the business records.</p> <p>Set the database name property to an existing database.</p>
GS-24209	<p>Invalid relation, primaryField and foreignField must have same number of fields.</p> <p>Modify the relation fields so that a primary field corresponds to each foreign field.</p>
GS-24210	<p>Invalid query, left and right join must have same number of columns.</p> <p>Check the query fields so that a left field corresponds to each right field.</p>
GS-24211	<p>Relation types don't match exactly.</p> <p>The type of a foreign key doesn't match the corresponding primary key's type.</p> <p>Check the relation, change the field types, or fix the primary / foreign key.</p>

Number	Description
GS-24212	<p>Relation field <i>fieldname</i> not found.</p> <p>The relation refers a field that is missing in the record.</p> <p>Modify the relation or add the field to the record.</p>
GS-24213	<p>Non existing schema <i>schema</i> attached to document.</p> <p>Check the schema being referenced.</p>
GS-24214	<p>Nonexistent table <i>table</i> referenced in document.</p> <p>The document uses a database table that is not present in the schema.</p> <p>Update the schema, or change the field using the table.</p>
GS-24215	<p>Nonexistent column <i>table.column</i> referenced in document.</p> <p>The document uses a database column that is not present in the schema.</p> <p>Update the schema or change the field using the <i>table.column</i>.</p>
GS-24216	<p>Database table defined in "no database" document.</p> <p>Set the database property to an existing schema or remove the field.</p>
GS-24217	<p>Table alias referenced more than once in the form.</p> <p>The same table alias is associated with two columns in the form.</p> <ul style="list-style-type: none"> • Change one table alias name. • Change either the table or column name.
GS-24218	<p>Alias <i>alias</i> referenced for different tables in document.</p> <p>The same alias is used for different tables.</p> <p>Rename the alias so that it refers to the same database table.</p>
GS-24219	<p>Alias Lookup field is ignored on %1 as it is the master table of the record.</p> <p>A lookup field is dedicated to foreign field update in the master table; do not set it on a master table column.</p> <p>Remove the lookup property value or update the field database settings.</p>
GS-24220	<p>Name value %1 is already used.</p> <p>Duplicate name used in the document.</p> <p>Rename the element so that the name is unique.</p>
GS-24221	<p>Invalid INTERVAL qualifier.</p> <p>The <i>qual1</i> or <i>qual2</i> set for the INTERVAL <i>sqlType</i> is not valid. It should belong to the INTERVAL classes (i.e., YEAR-MONTH or DAY-TIME)</p> <p>Change either <i>qual1</i> or <i>qual2</i> to fit the respective class range or change the <i>sqlType</i> property from INTERVAL to another <i>sqlType</i>.</p>
GS-24222	<p>Startfield of DATETIME or INTERVAL qualifiers must come earlier in the time-list than its endfield.</p>

Number	Description
	<p>The <i>qual1</i> value should be greater than the <i>qual2</i> value, when the <code>sqlType</code> is either DATETIME or INTERVAL.</p> <p>Ensure <i>qual1</i> is greater than <i>qual2</i> or change the <code>sqlType</code> property from INTERVAL to another <code>sqlType</code>.</p>
GS-24223	<p>Query properties set without attached database schema.</p> <p>Some query properties (join, order, additional tables or where) are defined without a database schema attached.</p> <p>Clear the query properties or attach a database schema to document.</p>
GS-24224	<p>Table <i>table</i> referenced by <i>property</i> is not present in the record.</p> <p>A table is referenced in a query property (join or order), but is not referenced in any record's field or additional tables.</p> <ul style="list-style-type: none"> • Remove the join or order that references this table. • Add a field referencing a column from this table to the record. • Add the table to the 'additional properties' property. • In the join or order, change the table to another one that is present in the record.
GS-24225	<p>Invalid relation, it must have at least one field.</p> <ul style="list-style-type: none"> • Remove the relation. • Add one or more source, destination field pair.
GS-24226	<p>Value not compatible with <code>dataType</code>.</p> <p>The value is not compatible with <code>dataType</code> set. This error occurs when <i>defaultValue</i> and <i>include</i> property value is incompatible with <code>dataType</code> property.</p> <p>Change <i>defaultValue/include</i>, enter a value compatible to <code>dataType</code> format.</p>
GS-24227	<p>Invalid join between %1 and %2.</p> <p>A join between the two tables is not correct (another one exists with a different operator).</p> <ul style="list-style-type: none"> • Remove the join • Change the operator • Change the join table(s)
GS-24228	<p>There is no join between table %1 and master table.</p> <p>The specified table is joined to another one, but not to the master table, creating a Cartesian product.</p> <ul style="list-style-type: none"> • Add the missing join for the %1 table • Remove the table %1
GS-24229	<p>Invalid initializer.</p> <p>The initializer format is incorrect.</p> <p>Change the initializer to respect the format.</p>
GS-24230	<p>Invalid source %1 for initializer %2.</p>

Number	Description
	<p>The initializer source (left of “.”) is unknown.</p> <p>Change the initializer to respect the format.</p>
GS-24231	<p>Initializer property %1 is missing.</p> <p>The property used in the initializer does not exist.</p> <p>Change the property name in the initializer.</p>
GS-24232	<p>Cannot resolve initializer value.</p> <p>The initializer cannot be resolved, the database element is not found, or the property is missing.</p> <p>Change the initializer to point to a valid element.</p>
GS-24233	<p>Orphan property %1, clean document settings to remove it.</p> <p>The document contains a dynamic property which is not present in the Application Generator template directory settings (orphan property). These orphan properties won't be taken into account during compilation. The Application Generator settings dynamic properties should match the document ones, otherwise this error is generated.</p> <ul style="list-style-type: none"> • Add the orphan properties to the <code>settings.agconf</code> file. • Remove the properties from the document using Tools >> Specific setup >> Clean orphan properties.
GS-24234	<p>Orphan property group %1.</p> <p>The document contains a dynamic property group which is not present in the Application Generator template directory settings (orphan property). The Application Generator settings dynamic properties should match the document ones, otherwise this error is generated.</p> <ul style="list-style-type: none"> • Add the orphan property groups to the <code>settings.agconf</code> file. • Remove the properties from the document using Tools >> Specific setup >> Clean orphan properties.
GS-24235	<p>File type not defined in Application Generator settings.</p> <p>A file used in Application Generator to generate code (<code>\$generate</code>) is of an undefined file type in <code>settings.agconf</code>.</p> <p>Add the item type definition to the <code>settings.agconf</code>.</p>
GS-24236	<p>Cannot resolve initializer without a valid database schema.</p> <p>The property initializer uses the database schema but the file doesn't have a schema.</p> <ul style="list-style-type: none"> • Set the database. • Set the property value so that the initializer is not resolved. • Remove the initializer.
GS-24237	<p>Unique key field %1 is not present in the Record.</p> <p>The unique key field value contains one field which is not in the record.</p> <ul style="list-style-type: none"> • Remove the field from the unique key.

Number	Description
	<ul style="list-style-type: none"> • Add the field to the record.
GS-24238	<p>Node %1 contains orphan properties, clean document settings to remove them.</p> <p>The file settings for the node differ from the Genero Studio ones, either the Genero Studio settings are not up to date, or the file contains old settings.</p> <ul style="list-style-type: none"> • Update the Genero Studio settings. • Clean the document settings (Tools >> Specific setup >> Clean orphan properties).
GS-24239	<p>Inactive table %1 in the database schema.</p> <p>The database table active flag is set to false, the table is inactive and cannot be used for code generation.</p> <p>Make the table active or do not use it.</p>

Business Application Diagram error messages

A list of Business Application Diagram error messages. For messages that are not self-explanatory, additional information is provided.

Table 77: Business Application Diagram Error Messages

Number	Description
GS-23001	Updated %1 property from %2 to %3.
GS-23402	<p>Item is conflicted.</p> <p>There are conflicts between items (they share same ID or file path).</p> <p>Resolve the conflict. (Right-click and select Resolve Conflict menu option.)</p>
GS-23403	<p>Item not implemented.</p> <p>Implement the item. (Right-click on item and select Implement.)</p>
GS-23405	<p>File %1 is missing.</p> <p>Some implemented item file is missing on the disk. Restore the file or recreate the item and implement it.</p>
GS-23406	<p>Item/Relation type %1 not supported.</p> <p>Open the template file settings.agconf and fix the errors.</p>
GS-23407	<p>Unsupported property %1 found for %2.</p> <p>Open the template file settings.agconf and add the property definition.</p>
GS-23408	<p>Duplicate Item name %1 found.</p> <p>Multiple items share the same name property and it should be unique. Modify name property value so that it is unique within the file items.</p>
GS-23409	<p>Duplicate Relation name %1 found.</p> <p>Multiple relations share the same name property, name should be unique. Modify name property value so that it is unique within the file relations.</p>

Number	Description
GS-23410	<p><Constraint> (<description>) constraint failed.</p> <p>Invalid source <Item>. When a constraint is defined with minSource and maxSource values as 0, it implies that the source item should not have any outgoing relations of type given in reference. This error occurs as the item has non-supported outgoing relations.</p> <p>Remove all the unsupported outgoing relations from the item.</p>
GS-23411	<p><Constraint> (<description>) constraint failed.</p> <p>Invalid destination <Item>. When a constraint is defined with minDestination and maxDestination values as 0, it implies that the destination item should not have any incoming relations of type given in reference. This error occur as the item has non-supported incoming relations.</p> <p>Remove all the unsupported incoming relations from the item.</p>
GS-23412	<p><Constraint> (<description>) constraint failed.</p> <p>Number of outgoing Relations, <Count>, are less than <Expected Count>.</p> <p>The numbers of outgoing relations from the item are less than the expected minimum outgoing relations count. Add at least <Expected Relation Count> number of outgoing relations to fix this problem.</p> <p>Number of outgoing Relations, <Count>, are greater than <Expected Count>.</p> <p>The numbers of outgoing relations from the item are greater than the expected maximum outgoing relation count. User has to remove the extra added relations, so that the relation count will not across <Expected Relation Count> which is the maximum number of outgoing relations allowed.</p> <p>Number of incoming Relations, <Count>, are less than <Expected Count>.</p> <p>The numbers of incoming relations to the item are less than the expected minimum incoming relations count. User has to add at least <Expected Relation Count> number of incoming relations to fix this problem.</p> <p>Number of incoming Relations, <Count>, are greater than <Expected Count>.</p> <p>The numbers of incoming relations to the item are greater than the expected maximum incoming relation count. User has to remove the extra added relations, so that the relation count will not across <Expected Relation Count> which is the maximum number of incoming relations allowed.</p>
GS-23413	<p>Item ID can not be blank.</p> <p>An ID is blank, the file is invalid. Contact your support center or remove the item containing the ID an create a new one.</p>
GS-23414	<p>Duplicate ID found for %1 and %2.</p> <p>Multiple items share the same id, creating a conflict. (Right-click and select Resolve Conflict menu option.)</p>
GS-23415	<p>%1 file not supported by Business Application Diagram %2.</p> <p>A build rule execution is trying to access data not described in the template settings.agconf (through the Business Application diagram). Update the template</p>

Number	Description
	settings.agconf file, adding the unsupported item or change the file build rule to not use the file.
GS-23416	Orphan property %1, clean document settings to remove it. Some defined property does not belong to the current settings (orphan) and are ignored by the compilation process. Check the metadata and either update the current template settings.agconf or clean orphan properties.
GS-23417	Orphan property group %1, clean document settings to remove it. Some defined property group does not belong to the current settings (orphan). Check the metadata and either update the current template settings.agconf or clean orphan properties
GS-23418	Node %1 contains orphan properties, clean document settings to remove them. Some node contains metadata definitions which do not belong to the current settings (orphan). They are ignored by the compilation process. Check the metadata and either update the current template settings.agconf or clean orphan properties.
GS-23419	Obsolete item type %1. The Item definition does not exist in the current template settings. It is ignore by the compilation process. Add the Item definition to the template settings.agconf, convert the item to a corresponding supported Item type (Right-click and select Convert menu option.), or remove the item.

Meta-schema Manager

The *Meta-schema Manager* is a visual tool used to design, create and maintain database meta-schema files.

- [What is a database meta-schema? \(4db\)](#) on page 288
- [Creating a meta-schema](#) on page 289
- [Adding more information to a meta-schema](#) on page 293
- [Viewing a meta-schema](#) on page 300
- [Update a meta-schema from database](#) on page 302
- [Generate a database script from meta-schema](#) on page 303
- [Meta-schema Manager Reference](#) on page 305

What is a database meta-schema? (4db)

A database meta-schema file is the central repository of a database's meta-data; information about the tables, columns, and relations, and default values of a relational database. Information from the database meta-schema file is used by Genero Studio Form Designer, Business Application Modeling, and Code Editor.

4db and 4dbx files

There are two types of database meta-schema files used by Genero Studio.

4db	Used in standard Genero Studio projects.
4dbx	Used in code-generated projects in Business Application Modeling. It includes special features for application generation such as a table for managing serial columns.

When is it used?

A database meta-schema file is required when you use Genero Studio to:

- Examine the structure of a tables, columns, and other attributes.
- Create a Genero Studio form definition based on a database table.
- Generate application code.

The database meta-schema is not used by Genero Studio when an application is executed. Genero Studio uses the Genero runtime system and the specific database client software to access any databases referenced in a Genero application.

Meta-schema creation and maintenance

Database meta data can be extracted from a relational database into a meta-schema file, or it can be based on an existing database schema file.

It is important that the database meta-schema file match the current structure of the database itself. If changes are made in the database structure, any database meta-schema files that describe the database must also be updated.

Meta-schema Manager includes options to create and update a database based on meta-schema file changes.

GSTSCHEMANAMES

Set the [GSTSCHEMANAMES](#) on page 143 environment variable to make meta-schemas available to all projects.

Creating a meta-schema

Information on creating meta-schemas.

- [Create a meta-schema](#) on page 227
- [Extract meta-schema information from database](#) on page 228
- [BDL schema file \(sch\)](#) on page 293
- [Add a meta-schema to a project](#) on page 231

Create a meta-schema

Create a database meta-schema file.

1. Select **File >> New**.
The **New** dialog opens.
2. Select a category from the left-side panel.

Option	Description
Genero BAM Desktop	Select this option if you are creating a database meta-schema for use in a Business Application Modeling managed project for a desktop application. The file created will be a 4dbx.
Genero BAM Mobile	Select this option if you are creating a database meta-schema for use in a Business Application Modeling managed project for a mobile application. The file created will be a 4dbx.
Genero	Select this option if you are creating a database meta-schema for use in a standard Genero program. The file created will be a 4db.

3. Select an option from the Database section from the right-side panel.

Option	Description
DB Schema	Creates a new meta-schema file.
DB Schema from Database	Opens the New Meta-schema dialog to which you enter your database and connection information. See Extract meta-schema information from database on page 228.

The meta-schema file displays in the document view. A new meta-schema file is automatically added to the list in the [DB Schemas Tab](#) if you have saved the file in the current project.

Extract meta-schema information from database

The **New Meta-schema** dialog assists in extracting schema information from a database.

To extract a database meta-schema file from a database, you must have access and permissions for the database. If you have trouble connecting to a database, make sure the database and the corresponding database client software are installed and configured properly.

When you extract the meta-schema information from the database, you overwrite the existing schema. Any user changes that had been made to the schema are lost when using the extract schema option. If you

wish to keep user changes, you must update the schema. See [Update a meta-schema from database](#) on page 302.

1. Select **Database>>Extract Schema**. The first step is specifying the name and location of the meta-schema file.

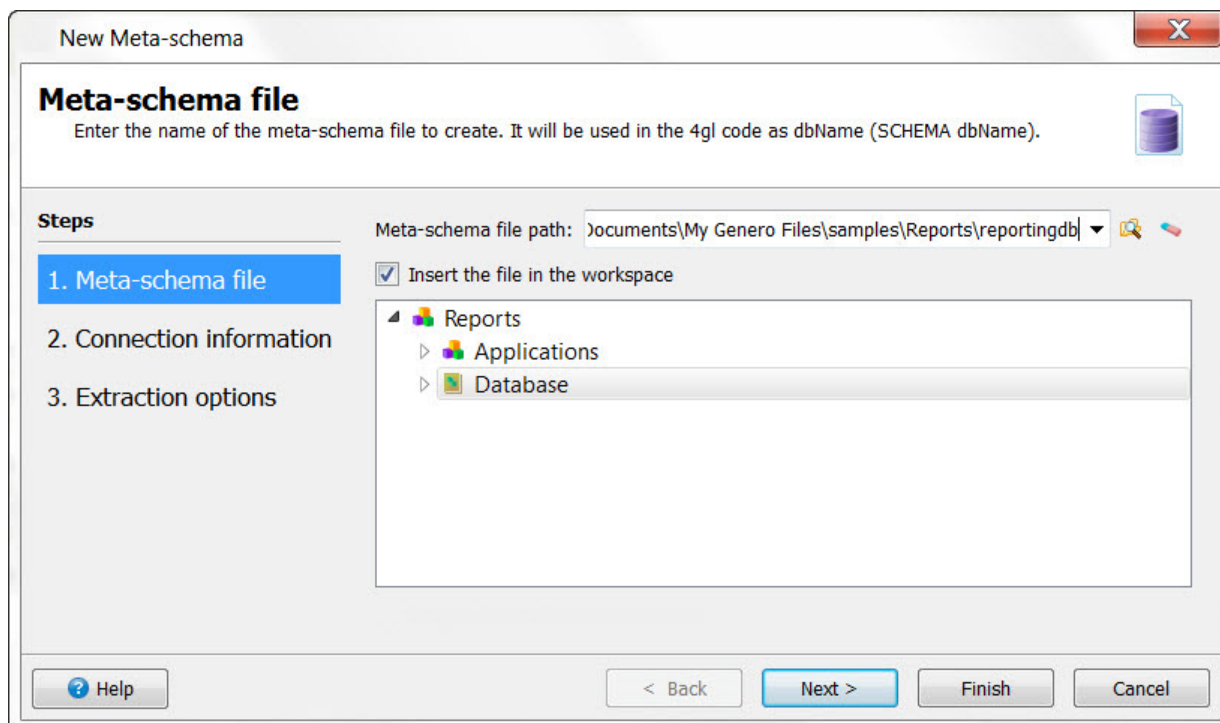


Figure 182: New Schema dialog

Meta-schema file path

Browse for a destination directory. Enter the name and path for the new database meta-schema file. Select a 4db meta-schema file for standard projects. Select a 4dbx file if you are working with a Business Application Modeling managed project.

Insert the file in the project

Check this box to add the meta-schema file in the project. Select the node where the file should be added.

2. Click the **Next** button to continue to **Connection information**. This connection information is only used to extract the information for the database meta-schema file from the referenced database.

New Meta-schema

Connection information
Enter the connection information required to connect to the database.

Steps

1. Meta-schema file
2. Connection information
3. Extraction options

Database Connection Information

Use explicit settings :

Previous connections :

Database type :

Database driver :

Informix server :

Database name :

Use external settings :

Extraction process relies on FGLPROFILE information to connect to the database. The provided schema name is used to connect to the database. Please refer to the Genero BDL manual for more information.

Database User Information

Table owner :

User name :

User password :

Figure 183: Connection information

- a) In the **Database Connection Information** section, select either **Use explicit settings** or **Use external settings**.

Use explicit settings, previous connection

You can use a **previous connection** that was created for the same database. The drop down list provides a list of the existing connections.

Use explicit settings, database type

You can enter the **Database Type** by selecting the desired type from the drop down list, and the corresponding information for that type. The **Database driver** for the database type is automatically entered. If other drivers exist, they are available in the drop down list.

Use external settings

Information in the `FGLPROFILE` configuration file is used to extract the corresponding connection information for the specified database. Genero Studio will use the schema name that you entered to check for any related entry in the `FGLPROFILE` configuration file, and will use those values to define the connection. See information on the `FGLPROFILE` file in the *BDL User Guide*.

- b) In the **Database User Information** section, provide the necessary database user details. The required information varies based on the database type selected. See [Database server/user information](#) on page 308.
- c) Click **Test Connection** to verify that the information is correct and that you are able to access the database.
3. Click the **Next** button to continue to **Extraction Options**. Select the options for the meta-schema file.

New Meta-schema

Extraction options
Select the extraction options you wish to use.

Steps

1. Meta-schema file
2. Connection information
3. Extraction options

Options

Case sensitivity :

Import system tables

Ignore errors

Conversion Method

Base type	Type A	Type B
BIGINT	<input checked="" type="checkbox"/> BIGINT	<input type="checkbox"/> DECIMAL(19,0)
BIGSERIAL	<input checked="" type="checkbox"/> BIGSERIAL	<input type="checkbox"/> DECIMAL(19,0)
BOOLEAN	<input checked="" type="checkbox"/> BOOLEAN (t=45)	<input type="checkbox"/> CHAR(1)
INT8	<input checked="" type="checkbox"/> INT8	<input type="checkbox"/> DECIMAL(19,0)
LVARCHAR(m)	<input checked="" type="checkbox"/> VARCHAR2(m)	<input type="checkbox"/> VARCHAR2(m)
SERIAL8	<input checked="" type="checkbox"/> SERIAL8	<input type="checkbox"/> DECIMAL(19,0)

Buttons:

Figure 184: Extraction options

Case sensitivity

Specify how case in database object names should be handled. **Case sensitive**: case won't be changed on database objects, **Lower case**: database object names will be converted to lower case, **Upper case** : database object names will be converted to upper case.

Import system tables

Check this box to include system tables in the schema.

Ignore errors

Specify that conversion errors should be ignored. If this option is unchecked, the extraction will stop as soon as an error occurs (for example, if a table column has an unsupported type.)

Conversion method

Select the type of conversion you wish for the specific data types; the default choice is Type A.

4. Click **Finish** to begin the extraction process.

If you didn't already do so, save the database meta-schema file in a node in the project; the database meta-schema will be added to the DB Schemas tab and made available to other modules.

Any application that uses the meta-schema file must have a dependency to the node where the meta-schema file was added. See [Add a meta-schema to a project](#) on page 231.

BDL schema file (sch)

Genero BDL `sch` files contain definitions of the database tables and columns.

This `sch` file is automatically created when you compile a Genero Studio meta-schema file, or you can import an existing one using the **Database >> Import SCH file** menu option.

Importing an existing Genero BDL column definition file (`sch`), converts the format to a Genero Studio meta-schema file (`4db`). You are prompted for the meta-schema name and path.

When the `sch` file is used

The BDL `sch` file is used when a Genero BDL module (`4gl` file) is compiled. If you compile a BDL module/program from the command line, you must have a copy of the `sch` file in the same directory or set `FGLDBPATH` environment variable specifying the directory in which the `sch` files can be found.

Add a meta-schema to a project

You may have to add the meta-schema file to a project.

If you used **File >> New** to create your project, the default structure of your project includes nodes for a **project**, **application**, **library**, and **databases**; the dependencies between the default nodes has been predefined. When you save your Meta-schema file in the Databases node of the project, the dependency for the application node in the project already exists.

However, if you created your own project structure, you must follow these steps.

1. Open the project.
2. Right-click on the application or library node in the Project view to which you want to add the meta-schema file and select **Add Files**. Locate and add the [meta-schema file](#).
3. Add a dependency to the Meta-schema file for any application or library nodes. Right-click the node and select [Advanced Properties, Dependencies](#). Check the box for the node containing the Meta-schema file.

Adding more information to a meta-schema

Add more information to meta-schemas, such as tables and columns, constraints, indexes, and foreign keys.

Warning: A rebuild of a project is not automatically done when the meta-schema file (`.4db`, `.4dbx`) is modified. It is the responsibility of the developer to recompile the appropriate parts of the project.

- [Add new tables and columns](#) on page 294
- [Add constraints or indexes](#) on page 294
- [Manage SERIALS](#) on page 300
- [Add foreign keys](#) on page 295
- [Centralize field information \(label, widget, default value\)](#) on page 300

Add new tables and columns

You can add tables and columns to a meta-schema.

Right-click in the background of the meta-schema diagram. Select **Add Table**. A new table with a single column is added to the diagram.

Right-click on the table and select **Add Column** to add an additional column. Repeat for as many columns as needed.

Set properties for the table and each column by selecting the item in the diagram or Structure view and editing its properties in the Properties view.

Add constraints or indexes

Constraints and indexes that are part of the database structure are displayed as part of the table, but additional constraints and indexes can be added.

Right-click the table in the meta-schema diagram and select **Add Constraint or Index**.

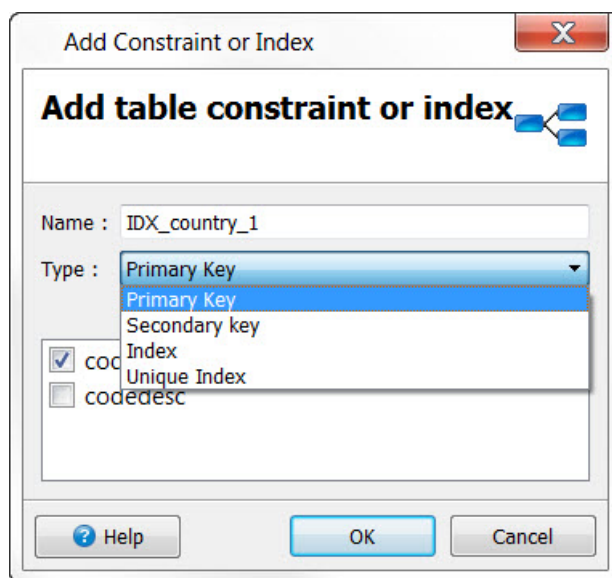


Figure 185: Add Constraint or Index dialog

Specify the index type: Primary key, Secondary key, Index, or Unique index.

Primary Key

A Primary Key is a column or set of columns that uniquely identifies a row of data. The Not Null property must be set on the columns used in a Primary Key. The Not Null property indicates the column does not allow NULL values. It is used by the generation and update scripts when generating the SQL statements used to manage the database.

Secondary Key

Also known as a Unique Constraint. Like Primary Key, the Secondary Key ensure uniqueness on the columns it is defined, but also allows NULL values.

Index

An index improves the speed of looking up data in tables. Indexes can be defined on one or more table columns.

Unique Index

Unique Index behaves the same as an Unique Constraint. It ensures the data in the column is unique. The difference between a Unique

Constraint and a Unique Index depends on the database engine.

The index can be viewed in the Structure view.

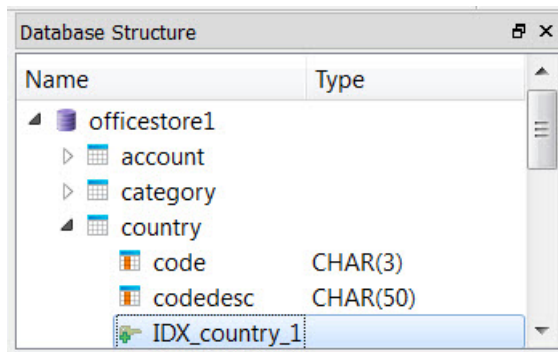


Figure 186: Database Structure view

Select the index to display its properties in the Properties view.

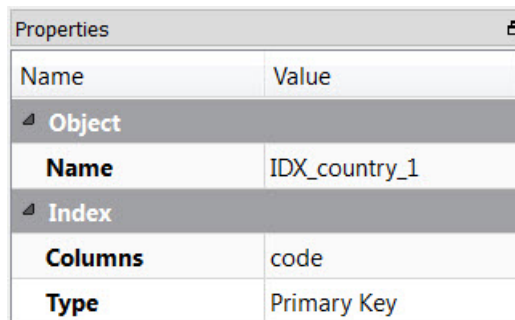


Figure 187: Properties view

If an index contains more than one column, the order of the columns is indicated.

Add foreign keys

A foreign key constraint specifies that values in one table must also appear in another table. Foreign keys that are part of the database structure are displayed, but foreign keys can also be added.

Right-click on the table in the meta-schema diagram to which you want to add a foreign key. Select **Add Foreign Key**.

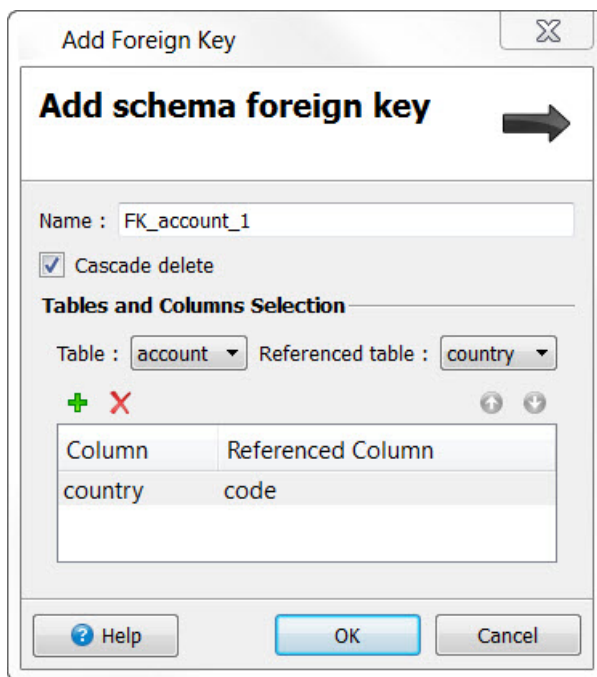


Figure 188: Add Foreign Key dialog

Name

Name of foreign key, a suggested name is provided. This name will display in the Structure and Properties views.

Table

Select the table that contains the foreign key.

Referenced table

Select the table that contains the primary key being referenced.

Column

Select the table column that references the primary key column in the referenced table.

Referenced Column

Select the name of the primary key column in the referenced table.

Cascade delete

Check this box if this is a foreign key with cascade delete. A foreign key with a cascade delete specifies that if a row in the parent table is deleted, then the corresponding rows in the child tables are automatically deleted. When this box is unchecked, the deletion of a row in a parent table will be aborted if a corresponding row exists in child tables. See [Cascade delete](#) on page 232.

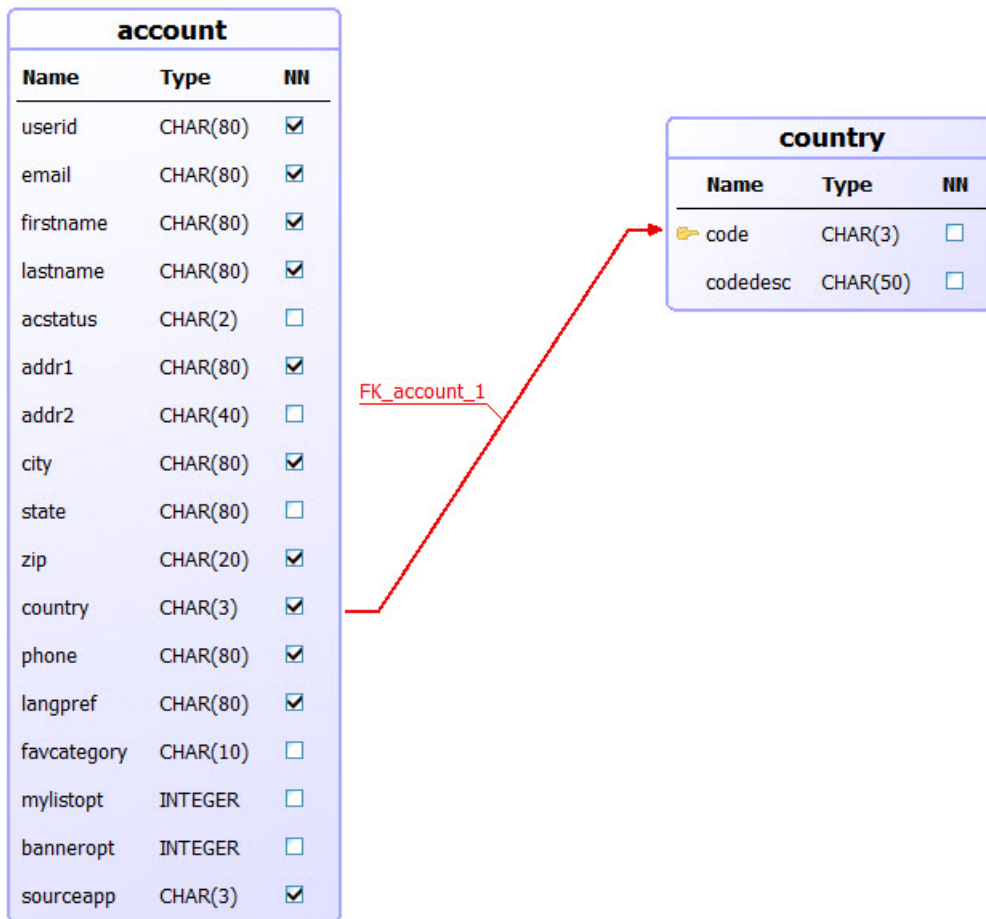


Figure 189: Foreign Key relationship in diagram

The foreign key constraint is added to the table in the Structure view. Select the foreign key constraint to display or edit its properties.

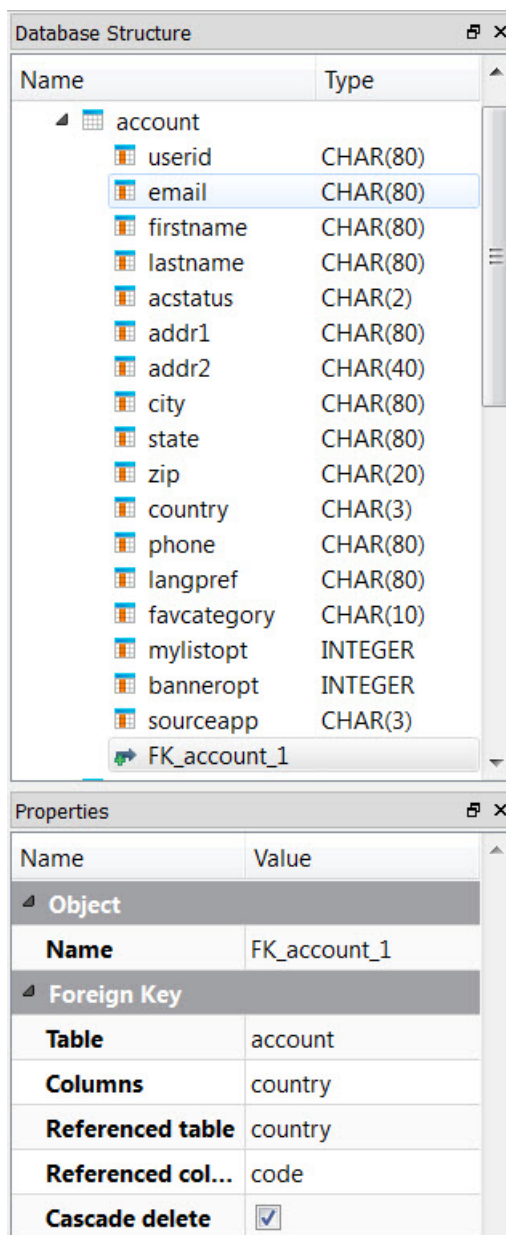


Figure 190: Foreign Key in Database Structure view

Add a foreign key by drawing relationship

You can also define the foreign key by drawing the relationship between columns using the mouse. Right-click on the background of the meta-schema diagram and select **Add Foreign Key**. Select the foreign key column in one table, and drag the mouse towards the primary key column in the table to be referenced. Once you release the mouse button, the relationship is displayed and the foreign key is added. Confirm the foreign key properties in the properties view.

Add a many-to-many relationship

To design a many-to-many relationship between two tables, create a junction table to link them together. Create the junction table using the primary key from each table.

Example

Table_A has a primary key column named Column_A.

Table_B has a primary key column named Column_B.

+ Table_A			
Name	Type	NN	
+ Column_A	CHAR(1)	<input checked="" type="checkbox"/>	
+ comment	CHAR(100)	<input type="checkbox"/>	

+ Table_B			
Name	Type	NN	
+ Column_B	CHAR(1)	<input checked="" type="checkbox"/>	
+ comment	CHAR(100)	<input type="checkbox"/>	

Figure 191: Initial tables, each with a primary key defined

To create the many-to-many relationship, add a junction table. This table is comprised of the primary key columns from each of the other tables.

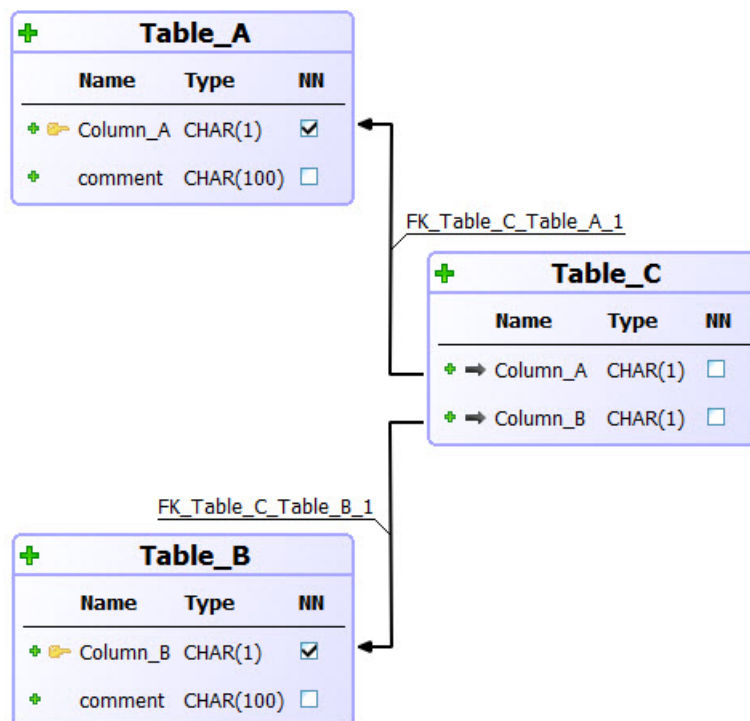


Figure 192: Create and include the junction table

Manage SERIALS

See "Auto-incremented columns (serials)" in the BDL User Guide.

Centralize field information (label, widget, default value)

Specifying field properties at the meta-schema level allows you to centralize properties such as the label, widget, and default value used when an application based on the meta-schema is built.

1. Open the meta-schema file (4db or 4dbx).
2. Select a column in the table and set its `Default value`, `label`, and/or `widget` properties.
For example, if you set the state field's `label` property to `State` and its `widget` property to `comboBox`, when you build a form that includes this field, it will be built with a label of `State` and as a `comboBox` instead of the default `Edit` widget type.

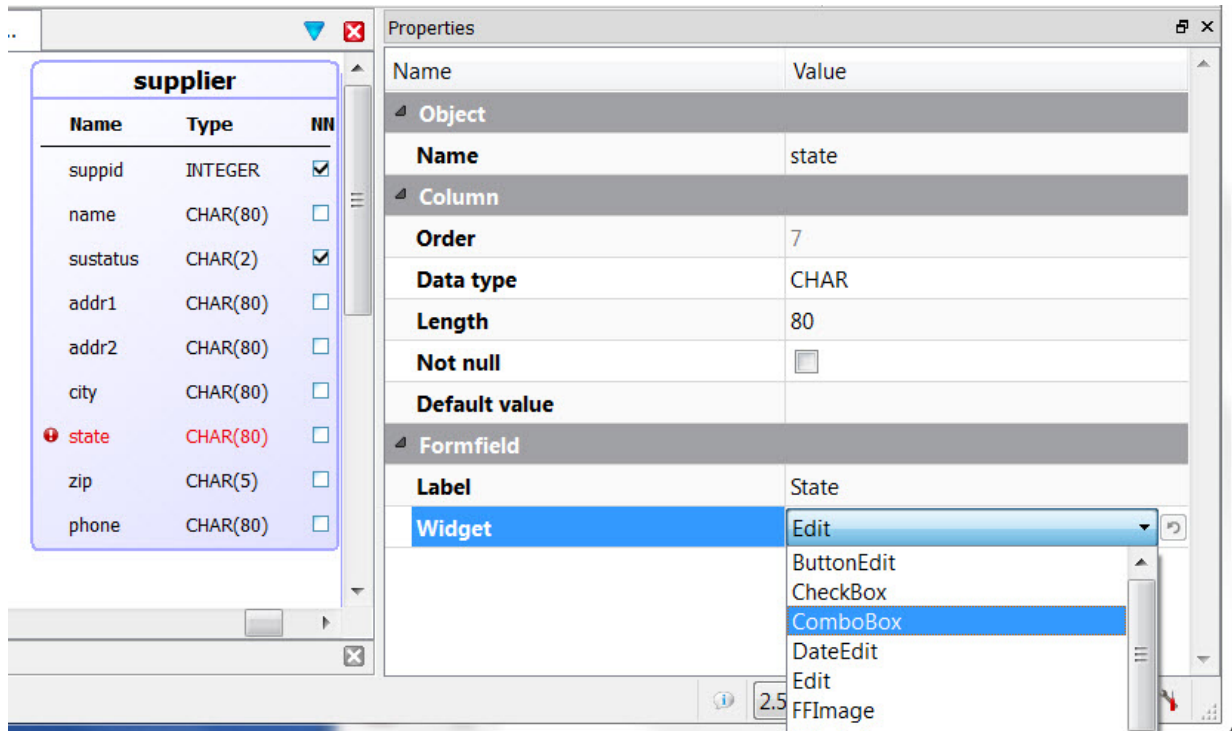


Figure 193: Setting properties at the meta-schema level

3. Repeat for all fields to which you want to centralize information.
4. Save the meta-schema file.

Viewing a meta-schema

Once a database meta-schema file has been created, it can be viewed and enriched with information that is not present in the database. Opening a meta-schema file displays it as a diagram for viewing and editing.

Each table is described by a table in the diagram. The structure of the database tables is displayed in the Database Structure view.

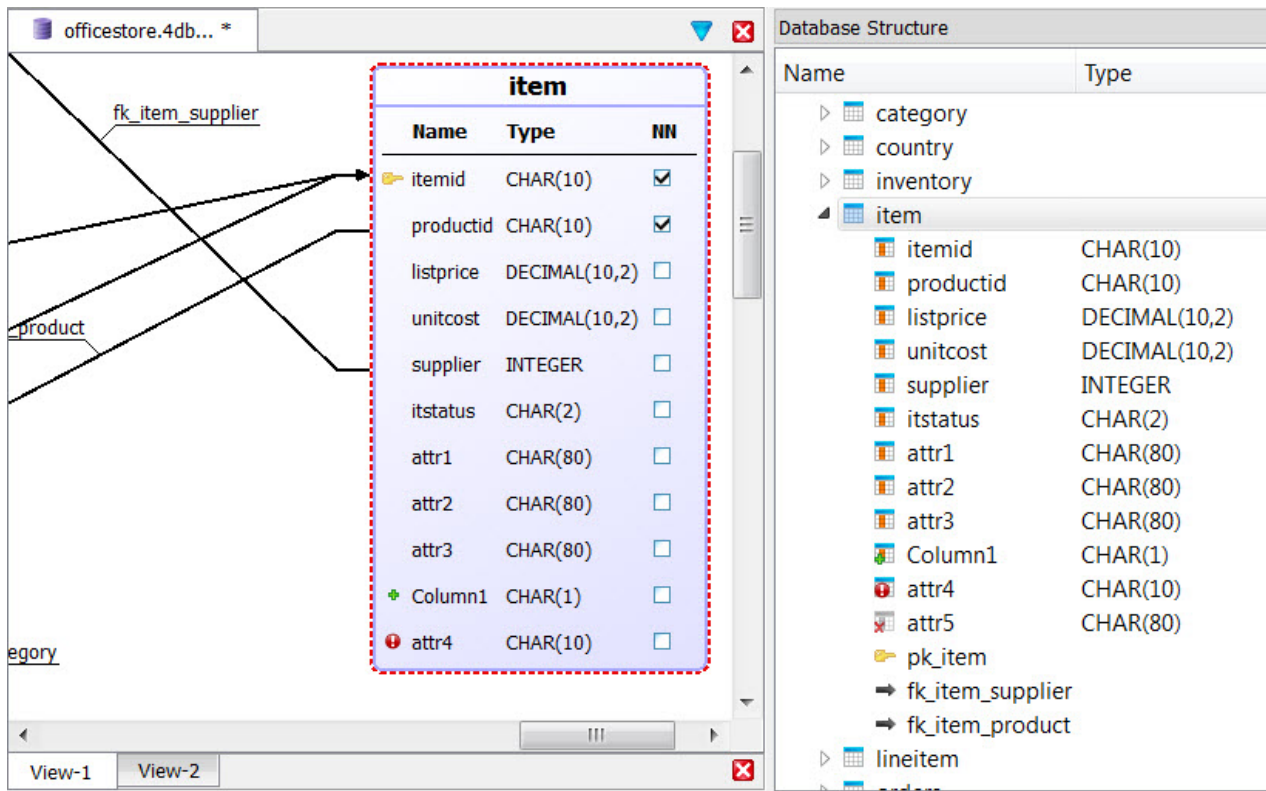


Figure 194: Viewing the schema

Icons on the diagram and in the Database Structure view indicate when a column's status has been changed:

- (+) modified
- (!) or removed
- (X)

A red circle icon with a white exclamation point, found in the upper left corner of the visual representation of the table, identifies a table that has been modified. To view the specifics of the modification, hover over the icon with your mouse.

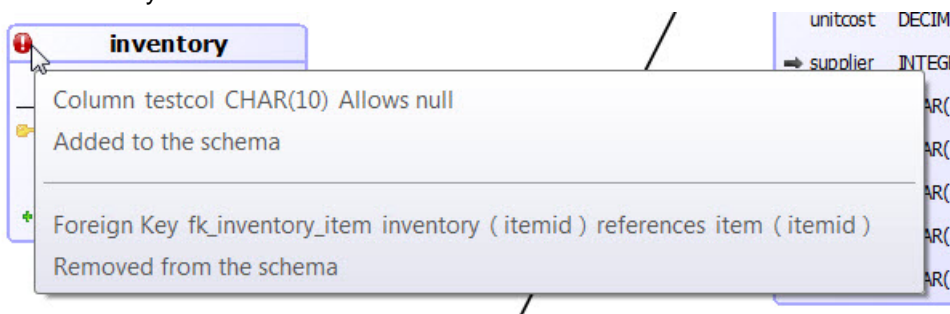


Figure 195: Viewing the modifications for a table

Zoom in and out

Use **Ctrl+mouse wheel** to zoom in and out on the diagram.

Reverting a change

Revert any changes made to the schema with **Database >> Revert....** See [Revert schema changes dialog](#) on page 315

Multiple views

Right-click the **View** tab at the bottom of the document to duplicate, rename, or delete a view. You may display multiple views of the same meta-schema.

Filter shown items

Filter the items shown on the diagram with the right-click context menu **Filter Items....**

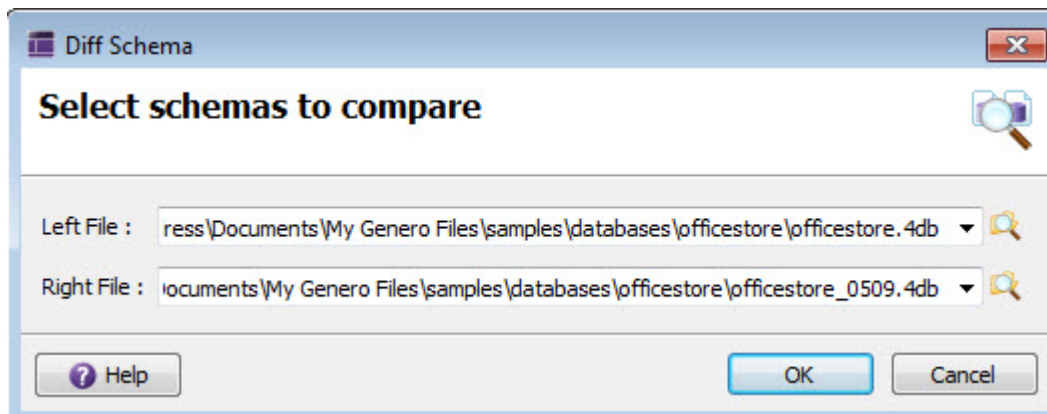
Comparing two meta-schemas

Compare two database meta-schema files.

Select **Database >> Diff Schema...** to create a merged schema view where:

- An object found in the first schema and not in the second will have a status of removed
✗
An object with a status of removed is not included in the schema diagram, but can be seen in the Database Structure view.
- An object found in the second schema and not in the first will have a status of added
+
- An object with the same name in both schemas but whose properties have changed in the second will have a status of modified
!

Figure 196: Selecting two meta-schema files to compare with Schema Diff



Update a meta-schema from database

Update a schema file to the current structure of its associated database.

This procedure merges any changes made in the structure of a database into your meta-schema file. This requires that you have access to the database. Any information that you have added to the meta-schema will be preserved.

When the changes are merged, the Meta-schema Manager verifies that the database objects in the original meta-schema still exist. If the object is no longer present in the database, it is removed from the meta-schema.

1. Right-click on the meta-schema file in the Projects or DB Schemas tab.

2. Select **Update Schema ...** or **Update from Database**.
3. Complete the **Update meta-schema** dialog as described in [Extract meta-schema information from database](#) on page 228.

The database meta-schema file is updated to match the current structure of the database.

Warning: A rebuild of a project is not automatically done when the meta-schema file (.4db, .4dbx) is modified. It is the responsibility of the developer to recompile the appropriate parts of the project.

Generate a database script from meta-schema

Generate a 4gl source file to be used to create or update a database that is described in the meta-schema file.

1. Right-click on the meta-schema file in the project.
2. Select an option:
 - **Generate Database Creation Script** to generate a source 4gl file that can be used to create a new database and tables according to the meta-schema file.
 - **Generate Database Update Script** to generate a source 4gl file that can be used to update an existing database based on the meta-schema file.

Note: The database update script will first drop the existing tables including their data, and then recreate the structure of the database based on the modified schema. Previous versions of the tables will be backed up and the data will be migrated to the new tables when applicable. It is recommended to perform a backup of the database prior to running the update script.
3. Complete the **Generate Database Script** dialog.

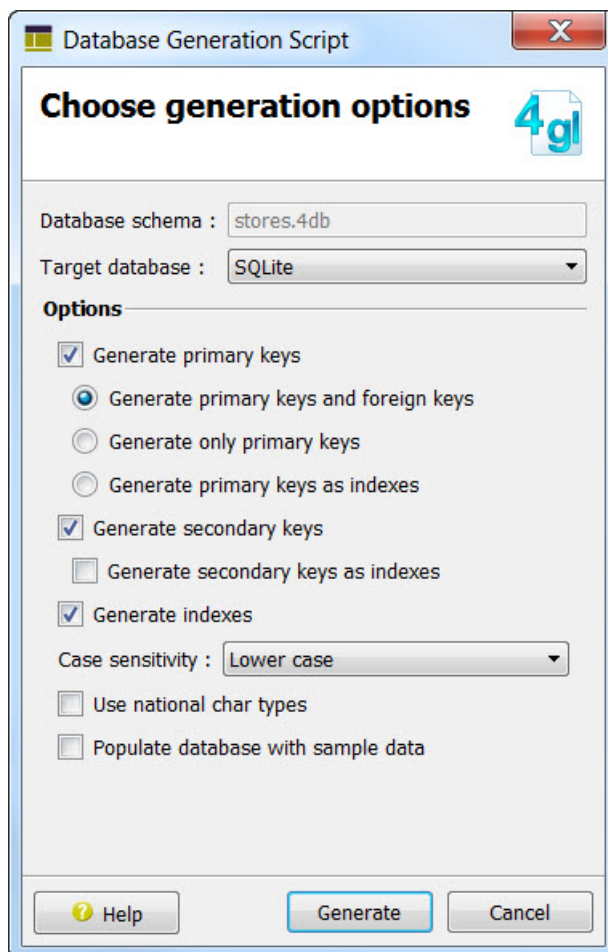


Figure 197: Generate Database Script dialog

Database schema

Name of selected meta-schema file.

Target database

Specify the database to use in the script.

Generate primary keys, secondary keys, indexes

Specify whether to include primary keys, secondary keys, and indexes in the script.

Use national char types

By default, the database creation / update scripts will generate column using standard char types. If this option is set, the scripts will produce columns using national char types. For example, with an Oracle database, the column types will be `CHAR` or `VARCHAR2` when the option is not selected, and `NCHAR`, `NVARCHAR2` when the option is selected.

Populate database with sample data

Add statements in the script to add sample data to the database.

4. Select **Generate**.

Generate meta-schema documentation

You can generate an HTML file that provides documentation on the meta-schema. It lists the tables, columns, indexes and foreign keys.

Any modifications made to the meta-schema will be considered as if applied to the database. New objects added to the meta-schema and modification made on existing objects will be seen in the documentation. Removed objects will not be shown.

1. Open a meta-schema file.
2. Select **Database >> Generate Schema Documentation**.
The documentation is generated in a temporary file and displayed in a browser.
3. To keep a copy of the documentation, save the page from the browser.

Meta-schema Manager Reference

Reference information for Meta-schema Manager.

- [Meta-schema properties](#) on page 305
- [Data types](#) on page 306
- [Database server/user information](#) on page 308
- [Dialogs](#) on page 309
- [Views](#) on page 315
- [GSTSCHEMANAMES](#) on page 143
- [Meta-schema diagram context menu](#) on page 318
- [Meta-schema Manager error messages](#) on page 318

Meta-schema properties

Properties can be set for each element in a meta-schema.

Table 78: Common properties

Property	Description
Name	The name property identifies the name of the item.

Table 79: Table properties

Property	Description
Active	The <code>active</code> property indicates that the table participates in the application code generation.

Table 80: Column properties

Property	Description
Order	Position of the column in the table.
Data Type	Specifies the data type of the column. See Data types on page 306.
Length	Defines the maximum length of the character string. The upper limit is 65534.
Precision	For <code>DECIMAL</code> data types; defines the number of significant digits (limit is 32, default is 16).

Property	Description
Scale	For <code>DECIMAL</code> data types; defines the number of digits to the right of the decimal point.
Qualifier 1, Qualifier 2	Specify the qualifiers for <code>INTERVAL</code> and <code>DATETIME</code> data typed columns, for example <code>YEAR</code> and <code>MONTH</code> .
Not null	Specifies that the column does not accept <code>NULL</code> values.
Default value	Assigns a default value to a column.
Label	Specifies the default label for a form item using the column.
Widget	Widgets are designed for data handling, action triggering, or decoration. Specify the default widget for a form item using the column.

Table 81: Index properties

Property	Description
Columns	The table columns that define the index.

Table 82: Foreign key properties

Property	Description
Table	Table that contains the foreign key.
Columns	The table column that references the primary key column in the referenced table.
Referenced table	The table that contains the primary key being referenced.
Referenced columns	The name of the primary key column in the referenced table.
Cascade delete	A foreign key with a cascade delete specifies that if a row in the parent table is deleted, then the corresponding rows in the child tables are automatically deleted.

Data types

Data types in a meta-schema have the same meaning for every supported database type.

If needed, the data type will be converted to the appropriate target database type when generating a database creation / update script. See the *SQL adaptation guides* in the *Genero Business Development Language User Guide* for more information on the use of data types in your database.

Table 83: Common data types

Type	Description
<code>BIGINT</code>	The <code>BIGINT</code> data type is used for storing very large whole numbers.
<code>BIGSERIAL</code>	The <code>BIG SERIAL</code> data type produces automatic integer sequences. <code>BIGSERIAL</code> is based on 64 bit integer sequences.
<code>BOOLEAN</code>	The <code>BOOLEAN</code> data type stores a logical value, <code>TRUE</code> or <code>FALSE</code> .
<code>BYTE</code>	The <code>BYTE</code> data type stores any type of binary data, such as images or sounds.

Type	Description
CHAR	The CHAR data type is a fixed-length character string data type.
DATE	The DATE data type stores calendar dates with a Year/Month/Day representation.
DATETIME	The DATETIME data type stores date and time data with time units from the year to fractions of a second.
DECIMAL	The DECIMAL data type is provided to handle large numeric values with exact decimal storage.
FLOAT	The FLOAT data type stores values as double-precision floating-point binary numbers with up to 16 significant digits.
INTEGER	The INTEGER data type is used for storing large whole numbers.
SERIAL	The SERIAL data type produces automatic integer sequences. SERIAL is based on 32 bit integer sequences.
SMALLFLOAT	The SMALLFLOAT data type stores values as single-precision floating-point binary numbers with up to 8 significant digits.
SMALLINT	The SMALLINT data type is used for storing small whole numbers.
TEXT	The TEXT data type stores large text data.
VARCHAR	The VARCHAR data type is a variable-length character string data type, with a maximum size. It is converted to the appropriate target database type when generating database creation / update script.

Table 84: Informix specific data types

Type	Description
INTERVAL	The INTERVAL data type stores spans of time as Year/Month or Day/Hour/Minute/Second/Fraction units.
MONEY	The MONEY data type is provided to store currency amounts with exact decimal storage.
NCHAR / NVARCHAR	IBM® Informix® supports the standard NCHAR and NVARCHAR data types. These types are equivalent to CHAR and VARCHAR (the same character set is used), except that the collation order is locale specific with NCHAR/ NVARCHAR types.

Informix specific types should be avoided when designing databases. A warning is displayed when a column uses an Informix specific type: INT8, INTERVAL, MONEY, NCHAR, NVARCHAR, NVARCHAR2, SERIAL8. To correct this warning convert columns to a common data type:

- INT8 to BIGINT
- MONEY to DECIMAL

- SERIAL8 to BIGSERIAL
- NCHAR / NVARCHAR / NVARCHAR2 to CHAR / VARCHAR and check the **Use national char types** option when generating database scripts
- avoid using INTERVAL

Note: The internal data type is used when generating database creation / update scripts or when working with the Business Application Modeler. However, if you edit a data type in a schema from the extracted database, the generic data type used in the creation of the schema might change the internal data type. For example, a column extracted from an Informix® database of type INT8 will be displayed in the schema as BIGINT. If the you change the type definition in the schema back to BIGINT, the database creation / update scripts will this time create a column of type BIGINT in the Informix® database.

Database server/user information

Database server and user information for each supported database type.

- [IBM DB2](#) on page 308
- [Informix](#) on page 308
- [MySQL](#) on page 308
- [Oracle](#) on page 309
- [PostgreSQL](#) on page 309
- [SQLServer](#) on page 309
- [SQLite](#) on page 309

IBM® DB2®

Table 85: IBM® DB2®

ODBC datasource	Name of the DataSource that has been previously created in ODBC.
User name	User name used for connection, or blank if not required.
User password	User password for the connection, or blank if not required.

Informix®

Table 86: Informix®

Informix® server	Name of the Informix® database server instance (the Informix® environment that contains the database to which you wish to connect). This is the same as the value of the INFORMIXSERVER variable.
User name	User name used for connection, or blank if not required.
User password	User password for the connection, or blank if not required.

Note: For Informix® SE, the system variable **DBPATH** must be set.

MySQL

Table 87: MySQL

Host name or IP address	Where the database is located.
Database name	MySQL database name.
User name	User name used for connection, or blank if not required.

User password	User password for the connection, or blank if not required.
---------------	---

Note: If you are using a remote MySQL database and a dynamic runner, these environment variables must be set in Configurations:

- `MYSQLDIR=mysql installation directory`
- `MYSQL_UNIX_PORT=$MYSQLDIR/mysql.sock`
- `MYSQLPORT=port used for mysql socket`

Oracle

Table 88: Oracle

TNS name	Oracle TNS service name.
Schema name	Name of Oracle schema.
User name	User name used for connection, or blank if not required.
User password	User password for the connection, or blank if not required.

PostgreSQL

Table 89: PostgreSQL

Host name or IP address	Where the database is located.
Port	Port number used to access the database host
Database name	PostgreSQL database name.
User name	User name used for connection, or blank if not required.
User password	User password for the connection, or blank if not required.

SQLServer

Table 90: SQLServer

ODBC datasource	Name of the DataSource that has been previously created in ODBC.
User name	User name used for connection, or blank if not required.
User password	User password for the connection, or blank if not required.

SQLite

Table 91: SQLite

Database file	Name of the file that contains the database.
User name	User name used for connection, or blank if not required.
User password	User password for the connection, or blank if not required.

Dialogs

Information about Meta-schema Manager dialogs.

- [Advanced Properties dialog](#) on page 310
- [Connection information dialog](#) on page 312
- [Database Generation Script dialog](#) on page 314

- [Revert schema changes dialog](#) on page 315

Advanced Properties dialog

The Advanced properties dialog provides options for extracting and generating database schema information.

The **Advanced Properties** dialog appears when you open a meta- schema (.4db, .4dbx) file and either:

- Right click on an empty space in the diagram and choose **Advanced properties** from the contextual menu
- Choose **Database >> Advanced properties**.

In addition:

- The form shown on the **Extraction** tab displays when you complete the process to extract a meta-schema from a database.
- The form shown on the **Generation** tab displays when you complete the process to generate a database script from a meta-schema file.

Extraction

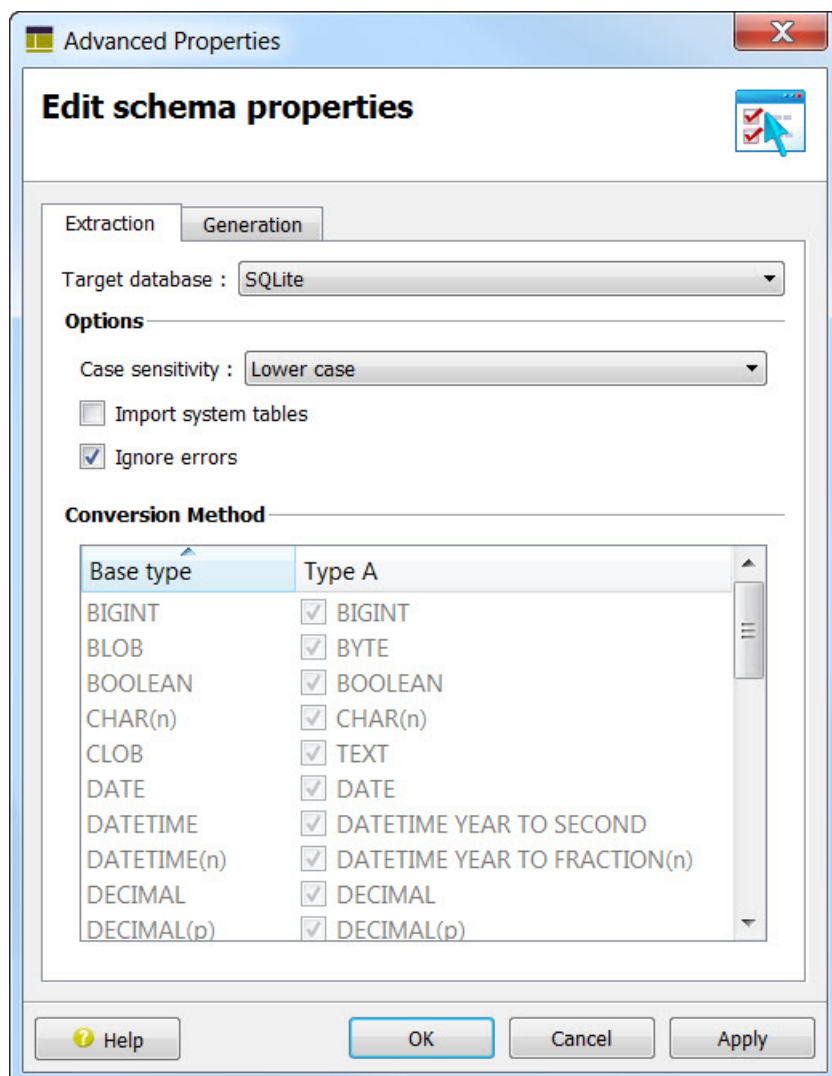


Figure 198: Advanced Properties dialog, Extraction tab

Case sensitivity

Specify how case in database object names should be handled. **Case sensitive**: case won't be changed on database objects, **Lower case**: database object names will be converted to lower case, **Upper case**: database object names will be converted to upper case.

Import system tables

Check this box to include system tables in the schema.

Ignore errors

Specify that conversion errors should be ignored. If this option is unchecked, the extraction will stop as soon as an error occurs (for example, if a table column has an unsupported type.)

Conversion method

Select the type of conversion you wish for the specific data types; the default choice is Type A.

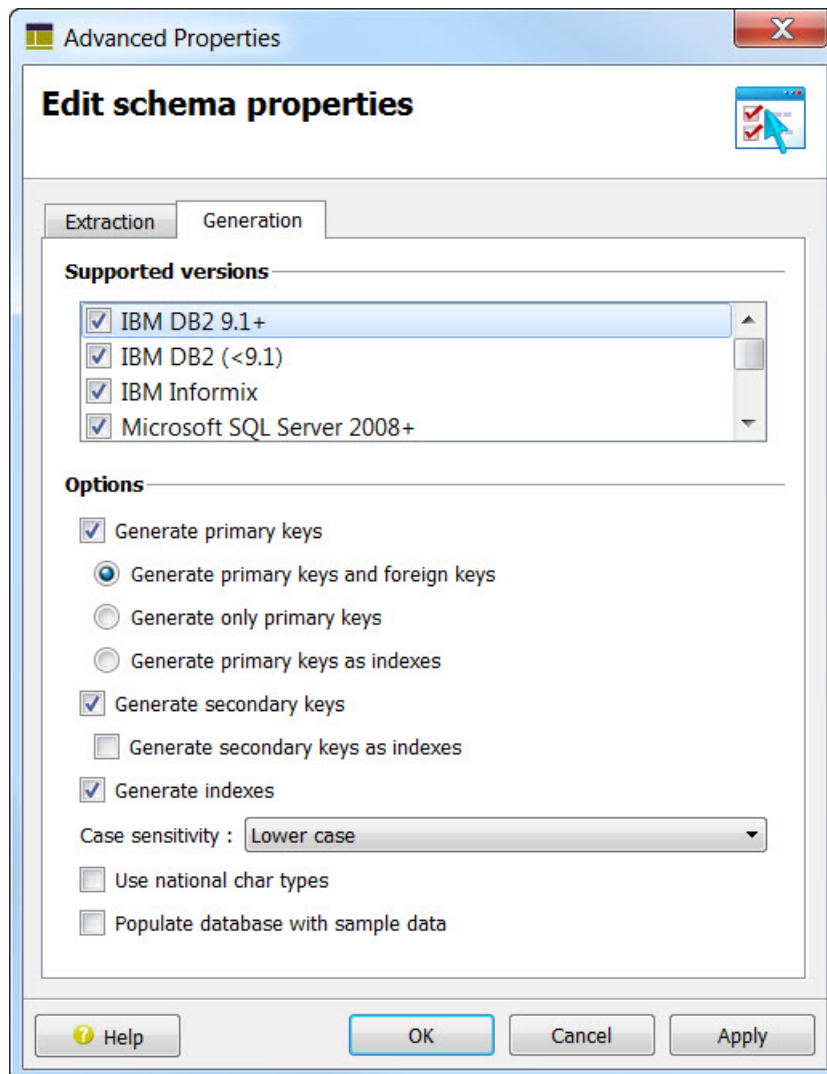
Generation

Figure 199: Advanced Properties dialog, Generation tab

Database schema

Name of selected meta-schema file.

Target database

Specify the database to use in the script.

Generate primary keys, secondary keys, indexes

Specify whether to include primary keys, secondary keys, and indexes in the script.

Use national char types

By default, the database creation / update scripts will generate column using standard char types. If this option is set, the scripts will produce columns using national char types. For example, with an Oracle database, the column types will be `CHAR` or `VARCHAR2` when the option is not selected, and `NCHAR`, `NVARCHAR2` when the option is selected.

Populate database with sample data

Add statements in the script to add sample data to the database.

Connection information dialog

The **Connection information** dialog gathers the details needed to connect to a database.

The **Connection information** dialog appears when you are creating or updating a meta-schema file, or when you are changing the connection for the DB Explorer plug-in.

Connection information

Connection information

Current schema : No schema selected

Keep schema connection information synchronized

Database Connection Information

Use explicit settings :

Previous connections :

Database type :

Database driver :

SQLite (Latest)

Database file :

Use external settings :

Extraction process relies on FGLPROFILE information to connect to the database. The provided schema name is used to connect to the database. Please refer to the Genero BDL manual for more information.

Database profile :

Database User Information

User name :

User password :

Test Connection

Help OK Cancel

Figure 200: Connection information dialog

The dialog can be seen as having four parts.

Schema details

The schema details section consists of two items. When not applicable, this section does not appear in the dialog.

Current schema

Identifies the current schema, if any. When you select to edit the database connection for a meta-schema file, the name of the currently connected database displays. If no connection has been made, the field states that no schema is selected.

Note: In some contexts, this read-only field does not display.

Keep schema connection information synchronized checkbox

When selected, the schema connection information will be permanently updated for the current meta-schema (.4db or .4dbx). This check box appears as selected and read-only when you edit the database connection for an existing meta-schema file. If you edit the connection from DB Explorer, you can uncheck this option to dissociate the DB Explorer connection information from the meta-schema connection information.

Note: In some contexts, this read-only field does not display.

Database Connection Information

In the **Database Connection Information** section, select either **Use explicit settings** or **Use external settings**.

Use explicit settings, previous connection

You can use a **previous connection** that was created for the same database. The drop down list provides a list of the existing connections.

Use explicit settings, database type

You can enter the **Database Type** by selecting the desired type from the drop down list, and the corresponding information for that type. The **Database driver** for the database type is automatically entered. If other drivers exist, they are available in the drop down list.

Use external settings

Information in the FGLPROFILE configuration file is used to extract the corresponding connection information for the specified database. Genero Studio will use the schema name that you entered to check for any related entry in the FGLPROFILE configuration file, and will use those values to define the connection. See information on the FGLPROFILE file in the *BDL User Guide*.

Database User Information

In the **Database User Information** section, provide the necessary database user details. The required information varies based on the database type selected. See [Database server/user information](#) on page 308.

Test Connection

Click **Test Connection** to verify that the information is correct and that you are able to access the database.

Database Generation Script dialog

Generate a 4gl source file to create or update the database that is described in the meta-schema file.

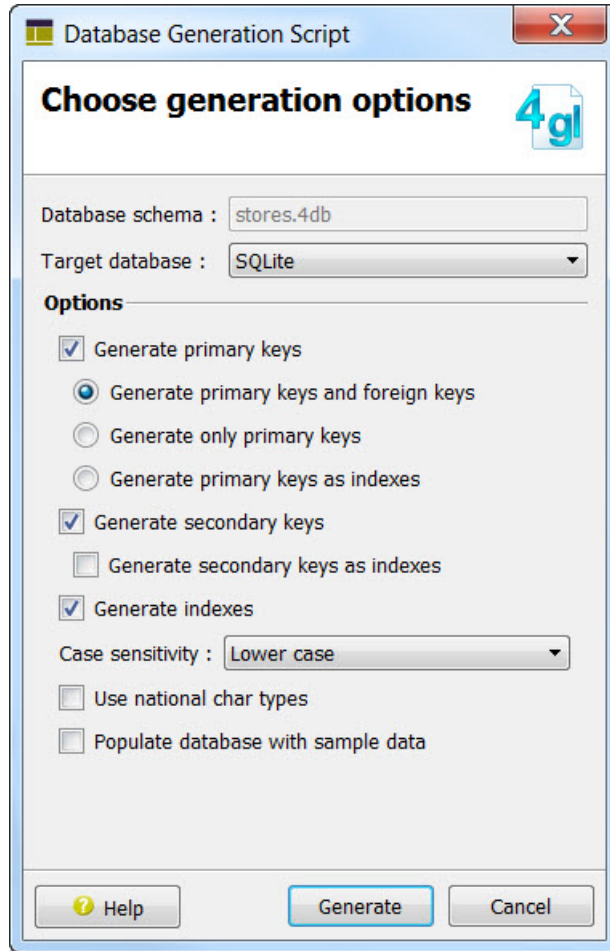


Figure 201: Database Generation Script dialog

Database schema

Name of selected meta-schema file.

Target database

Specify the database to use in the script.

Generate primary keys, secondary keys, indexes

Specify whether to include primary keys, secondary keys, and indexes in the script.

Use national char types

By default, the database creation / update scripts will generate column using standard char types. If this option is set, the scripts will produce columns using national char types. For example, with an Oracle database, the column types will be `CHAR` or `VARCHAR2` when the option is not selected, and `NCHAR`, `NVARCHAR2` when the option is selected.

Populate database with sample data

Add statements in the script to add sample data to the database.

Revert schema changes dialog

Changes made to the meta-schema can be reverted with the **Revert schema changes** dialog.

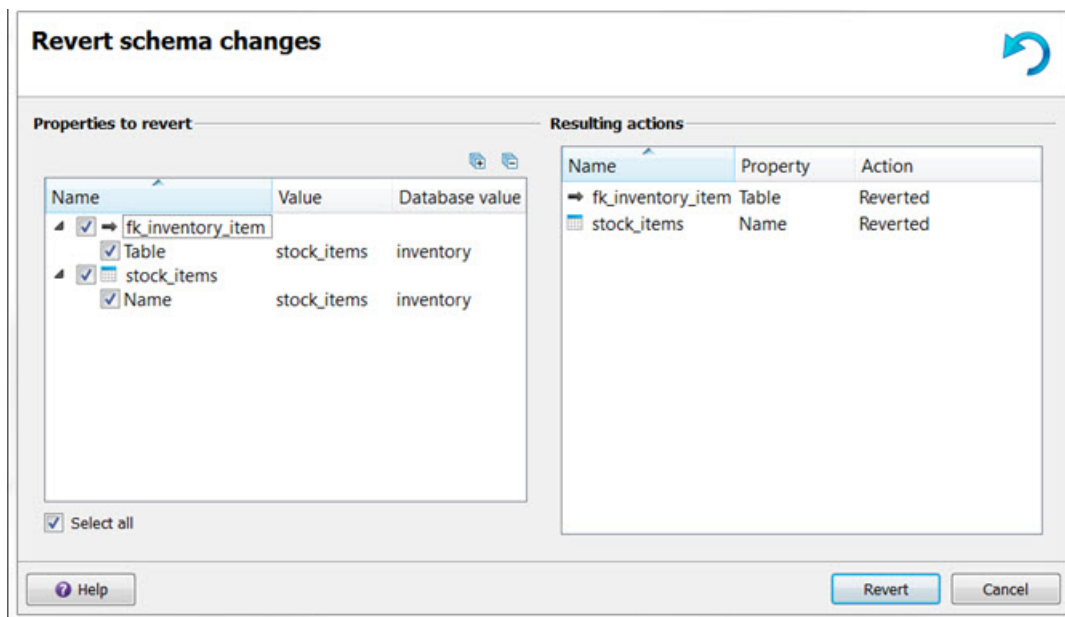


Figure 202: Revert changes dialog

Properties to revert

Lists all elements in the meta-schema that have been modified. Check the elements you wish to restore to its status when the schema was last extracted.

Resulting actions

Lists dependencies affected by the modified elements.

Views

Information about Meta-schema Manager views.

- [Filter view](#) on page 317
- [DB Schemas tab](#) on page 316

DB Schemas tab

The DB Schemas tab displays all database meta-schema files associated with the project.

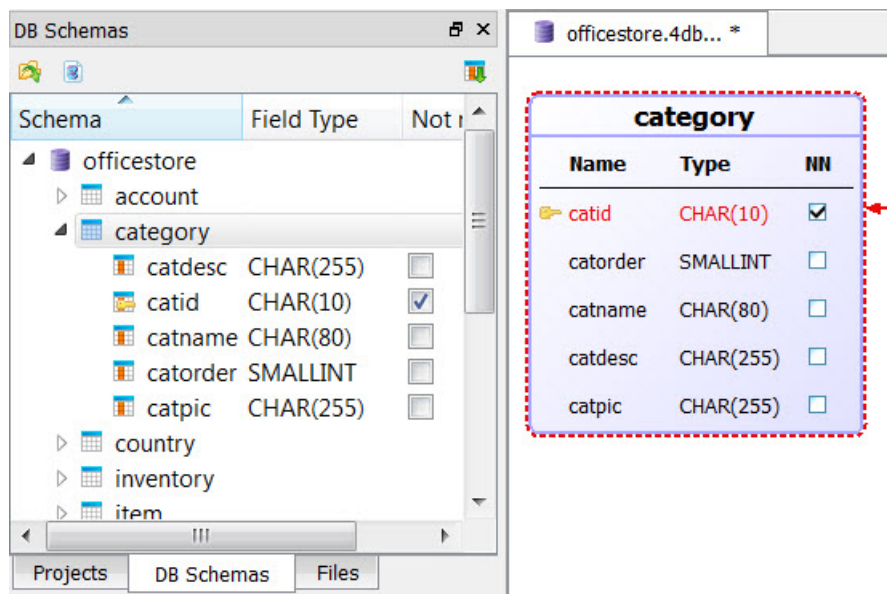


Figure 203: DB Schemas tab

A database meta-schema is added to the DB Schemas tab when you:

- Save a new database meta-schema file (4db or 4dbx) in the current project; the Meta-schema is available in the DB Schemas tab when you work on projects in that workspace.
- Set the [GSTSCHEMANAMES](#) on page 143 environment variable; the available meta-schemas are available in the DB Schemas tab for all projects.

Use the integrated Toolbar to:

Open a meta-schema

Open the meta-schema in Meta-schema Manager.

Update from database

See [Update a meta-schema from database](#) on page 302.

Sort column in database order

Change the sort order of the columns, from alphabetical order (the default) to the order in which the columns are defined in the database.

Filter view

The Filter View dialog allows you to hide and show items on a diagram.

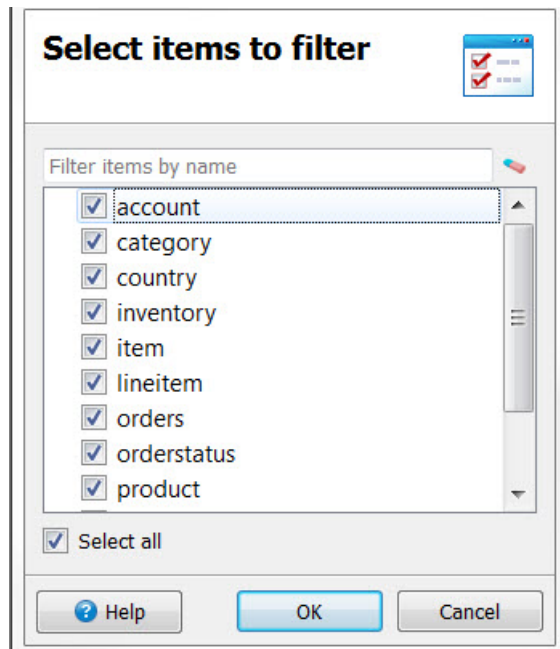


Figure 204: Filter View

Meta-Schema Manager preferences

Set default preferences for Meta-Schema Manager and DB Explorer.

DB Explorer

Table 92: DB Explorer options

Preference	Description
Maximum number of results for show/edit data field	Limit the number of rows returned by DB Explorer. When set to zero, unlimited rows can be returned.
Verbose mode check box	<p>When checked, information about the executed query displays.</p> <p>For a successful query execution, a message like the following displays: Executed query: select userid, password from signon</p> <p>If an error occurs, information about the SQL error displays. For example, if you edit the query and provide a wrong column name, an error message like the following displays: error (-6372): near "wrong_userid": syntax error Connection id: C:/Users/name/xxx/officestore.db Executed query: insert wrong_userid, password from signon</p>

Meta-schema diagram context menu

Right-click on a meta-schema diagram or item in the Database Structure view for a context menu of options.

Table 93: Meta-schema diagram context menu

Option	Description
Add Table	See Add new tables and columns on page 294.
Add Column	See Add new tables and columns on page 294.
Add Constraint or Index	See Add constraints or indexes on page 294.
Edit Constraint or Index	See Add constraints or indexes on page 294.
Add Foreign Key	See Add foreign keys on page 295.
Edit Foreign Key	See Add foreign keys on page 295.
Edit	Edit properties of the item selected.
Insert Column Before/After	See Add new tables and columns on page 294.
Revert	See Revert schema changes dialog on page 315.
Layout	Rearrange the items in the diagram.
Advanced Properties	Specify the extraction and/or generation options. See Advanced Properties dialog on page 310.
Filter Items...	The Filter View dialog allows you to hide and show items on a diagram.
Locate in Diagram	This action brings focus in the diagram to the selected item. If the selected object is not visible in the current view, the Meta-schema Manager will try to find another view where the object is visible. If no view is found, you are prompted to make the object visible in the current view or to create a new view.

Meta-schema Manager error messages

A list of Meta-schema Manager error messages. For messages that are not self-explanatory, additional information is provided.

Table 94: Meta-schema Manager Error Messages

Number	Description
GS-11000	Unexpected error. An unexpected error occurred.
GS-11001	Error while reading <i>file</i> schema: <i>description</i> . An error occurred while reading the file <i>file</i> . The details about the error can be found in the <i>description</i> part of the message.
GS-11002	Cannot find codec. The appropriate codec cannot be found to read a file.

Number	Description
GS-11003	<p>Cannot find codec for <i>encoding</i> encoding.</p> <p>The appropriate codec for encoding <i>encoding</i> cannot be found.</p>
GS-11004	<p>Error while creating new schema document.</p> <p>The Meta-schema Manager failed to create a new document.</p>
GS-11005	<p>Dynamic property <i>property</i> not found for node <i>node</i>.</p> <p>A dynamic property used in the meta-schema is not found in the current template. Check the appropriate template is selected before opening the meta-schema.</p> <p>Failed to read dynamic property.</p> <p>An error occurred while reading dynamic properties. The meta-schema file might be corrupted.</p>
GS-11050	<p>Schema <i>file</i> defined in GSTSCHEMANAMES cannot be found in FGLDBPATH.</p> <p>The file cannot be found.</p>
GS-11051	<p>Failed to load file <i>file</i>.</p> <p>An error occurred while loading the file <i>file</i>.</p>
GS-11052	<p>Failed to open file <i>file</i> for writing.</p> <p>Check the file permissions and those of its owner directory.</p>
GS-11053	<p>Failed to open file <i>file</i> for reading.</p> <p>Check the file permissions and those of its owner directory.</p>
GS-11100	<p>Failed to locate node.</p> <p>The specified database object cannot be found in the schema.</p>
GS-11101	<p>Unexpected schema.</p> <p>An unexpected schema object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</p>
GS-11102	<p>Missing schema name.</p> <p>The schema name cannot be found. This property is required and the loading of the schema file cannot continue.</p>
GS-11103	<p>Missing schema property <i>property</i>.</p> <p>This property is required, but it cannot be found. The loading of the schema file cannot continue.</p>
GS-11104	<p>Invalid schema property <i>property</i>.</p> <p>The schema property <i>property</i> has an invalid value. The loading of the schema file cannot continue.</p>
GS-11105	<p>Schema contains no table or column.</p>

Number	Description
	No valid tables have been found in the schema. A valid table is a table in which <i>is not flagged</i> has been removed and which contains at least one column in which <i>is not flagged</i> has been removed.
GS-11106	Empty schema file. The schema document contains no information.
GS-11107	File is saved using a newer version of Genero Studio. Please upgrade Genero Studio. The schema document has been saved with a newer version of Genero Studio. You will need to upgrade Genero Studio to be able to open it.
GS-11108	Schema name is not lowercase, its use in 4GL source will generate errors. The schema name contains uppercase characters. Using it in 4GL programs might generate compilation errors. Use lowercase for the Schema name.
GS-11109	The usage of both CHAR/VARCHAR and NCHAR/NVARCHAR columns is not supported by all databases. The schema contains columns with mutually exclusive types. For instance it contains both CHAR / VARCHAR and NCHAR / NVARCHAR columns.
GS-11110	Schema name is invalid, its use in 4GL source will generate errors. The schema name contains characters not supported by the Genero language.
GS-11111	Unexpected table. An unexpected table object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.
GS-11112	Missing table name. The table name cannot be found. This property is required and the loading of the schema file cannot continue.
GS-11113	Missing table property <i>property</i> . The table property <i>property</i> is required but it cannot be found. The loading of the schema file cannot continue.
GS-11114	Invalid table property <i>property</i> . The table property <i>property</i> has an invalid value. The loading of the schema file cannot continue.
GS-11115	Schema already contains a table <i>table</i> . The table has already been found in the 4db or 4dbx document. The document cannot contain two tables with the same name.
GS-11116	Table <i>table</i> is not found. Creating temporary table.

Number	Description
	The table cannot be found, and it is required by another database object. A surrogate table will be created.
GS-11117	Table contains no columns. The table does not contain any columns.
GS-11121	Unexpected column. An unexpected column object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.
GS-11122	Missing column name. The column name cannot be found. This property is required and the loading of the schema file cannot continue.
GS-11123	Missing column property <i>property</i> . The column property <i>property</i> is required but it cannot be found. The loading of the schema file cannot continue.
GS-11124	Table <i>table</i> already contains a column <i>column</i> . The column has already been found in the table. The document cannot contain two columns with the same name within the same table.
GS-11125	Constraint or index <i>index</i> already contains a column <i>column</i> . The column has already been found in the index. The document cannot contain two columns with the same name within the same index.
GS-11126	Foreign key <i>key</i> already contains a column <i>column</i> . The document cannot contain two columns with the same name within the same foreign key.
GS-11127	Column <i>column</i> is not found in table <i>table</i> . Creating temporary column. The column cannot be found in the table and it is required by another database object. A surrogate column will be created.
GS-11128	Column <i>column</i> in table <i>table</i> has unsupported type: <i>type</i> . The type is not supported for column within table. An unsupported type is of form encoded=(type, length), and can be generated while extracting database schemas.
GS-11129	Column <i>column</i> in table <i>table</i> has invalid type: <i>type</i> . The type is invalid for column within table. An invalid type is not handled by the 4GL language.
GS-11131	Unexpected foreign key. An unexpected foreign key object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.
GS-11132	Missing foreign key name.

Number	Description
	The foreign key name cannot be found. This property is required, and the loading of the schema file cannot continue.
GS-11133	Missing foreign key property <i>property</i> . The foreign key property <i>property</i> is required, but it cannot be found. The loading of the schema file cannot continue.
GS-11134	Schema already contains a foreign key <i>key</i> . The foreign key has already been found in the 4db or 4dbx document. The document cannot contain two foreign keys with the same name.
GS-11135	Schema already contains a foreign key <i>key</i> . Renaming user-defined foreign key to <i>name</i> . The foreign key has already been found in the 4db or 4dbx document. The document cannot contain two foreign keys with the same name. The foreign key will be renamed to <i>name</i> .
GS-11136	Foreign key <i>key</i> is not found. Creating temporary foreign key. The foreign key <i>key</i> cannot be found, and it is required by another database object. A surrogate foreign key will be created.
GS-11137	Table <i>table</i> is not found for foreign key <i>key</i> . The table the foreign key refers to cannot be found. The loading of the schema file cannot continue.
GS-11138	Column <i>column</i> is not found in table <i>table</i> for foreign key <i>key</i> . The column the foreign key refers to cannot be found in the table. The loading of the schema file cannot continue.
GS-11139	Foreign key <i>key</i> does not contain any table column. The foreign key does not contain any columns.
GS-11140	Foreign key <i>key</i> columns <i>column</i> and <i>reference</i> have different data types. The foreign key columns have mismatching data types.
GS-11141	Unexpected constraint or index. An unexpected index object has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.
GS-11142	Missing constraint or index name. The index name cannot be found. This property is required and the loading of the schema file cannot continue.
GS-11143	Missing constraint or index property <i>property</i> . The index property is required but it cannot be found. The loading of the schema file cannot continue.

Number	Description
GS-11144	<p>Table <i>table</i> already contains a constraint or index named <i>index</i>.</p> <p>The index has already been found in the table <i>table</i>. The document cannot contain two indexes with the same name within the same table.</p>
GS-11145	<p>Table <i>table</i> already contains a constraint or index named <i>index</i>. Renaming user-defined constraint or index to <i>name</i>.</p> <p>The index has already been found in the table. The document cannot contain two indexes with the same name within the same table. The index will be renamed to <i>name</i>.</p>
GS-11146	<p>Constraint or index <i>index</i> is not found. Creating temporary constraint or index.</p> <p>The index cannot be found in the table and it is required by another database object. A surrogate index will be created.</p>
GS-11147	<p>Column <i>column</i> is not found in table <i>table</i> for constraint or index <i>index</i>.</p> <p>The column that the index refers to cannot be found in the table. The loading of the schema file cannot continue.</p>
GS-11148	<p>Table <i>table</i> contains more than one primary key.</p> <p>The table contains more than one primary index. Only one primary index can be defined for a table.</p>
GS-11149	<p>Constraint or index <i>index</i> does not contain any table column.</p> <p>The index does not contain any columns.</p>
GS-11150	<p>Column <i>column</i> allows null values but is referred by primary key <i>primary key</i>.</p> <p>A column referred in a primary key doesn't have its NOT NULL flag set.</p>
GS-11181	<p>Unexpected layout data section.</p> <p>An unexpected layout section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</p>
GS-11182	<p>Missing layout data property <i>property</i>.</p> <p>The layout property <i>property</i> is required but it cannot be found. The loading of the schema file cannot continue.</p>
GS-11191	<p>Unexpected database data section.</p> <p>An unexpected data section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</p>
GS-11192	<p>Missing database data property <i>property</i>.</p> <p>The data property <i>property</i> is required but it cannot be found. The loading of the schema file cannot continue.</p>
GS-11193	<p>Document already contains database value for <i>identifier</i>.</p>

Number	Description
	Database data has already been found in the 4db or 4dbx document. The document cannot contain twice data for the same identifier.
GS-11194	<p>Document contains invalid identifier <i>identifier</i> for database value (value: <i>value</i>). Database value will be ignored and removed.</p> <p>The database data identifier doesn't match any database object. The associated value is ignored and will be discarded upon save.</p>
GS-11201	<p>Unexpected dynamic property.</p> <p>An unexpected dynamic property has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</p>
GS-11202	<p>Missing dynamic property attribute <i>attribute</i>.</p> <p>The dynamic property attribute <i>attribute</i> is required but it cannot be found. The loading of the schema file cannot continue.</p>
GS-11203	<p>Object doesn't support dynamic property <i>property</i>.</p> <p>A dynamic property has been read for a database object but this property is not defined for this object. Check the appropriate template is selected before opening the meta-schema.</p>
GS-11204	<p>Object already contains a dynamic property <i>property</i>.</p> <p>A dynamic property has been defined twice for a database object.</p>
GS-11210	<p>Invalid model node info version.</p> <p>Model node info for type <i>type</i> not found.</p> <p>Failed to add property <i>property</i> to model node info: a dynamic property with this name already exists</p> <p>Failed to add property <i>property</i> to model node info: a static property with this name already exists</p> <p>Failed to add property <i>property</i> to model node info: a property with this name was not found</p> <p>Information about model node is invalid.</p>
GS-11220	<p>Name cannot be empty.</p> <p>The name of a database object is empty.</p>
GS-11221	<p>Same name cannot be applied to more than one table.</p> <p>Two tables cannot have the same name in the meta-schema.</p>
GS-11222	<p>Same name cannot be applied to more than one column of same table.</p> <p>Two columns of the same table cannot have the same name in the meta-schema.</p>
GS-11223	<p>Same name cannot be applied to more than one foreign key.</p>

Number	Description
	Two foreign keys cannot have the same name in the meta-schema.
GS-11224	<p>Same name cannot be applied to more than one constraint or index of same table.</p> <p>Two constraints or indexes of the same table cannot have the same name in the meta-schema.</p>
GS-11225	<p>Table name is invalid.</p> <p>A table cannot be named with the provided name.</p>
GS-11226	<p>Column name is invalid.</p> <p>A column cannot be named with the provided name.</p>
GS-11227	<p>Database object name contains invalid character(s).</p> <p>The name of the database object contains invalid characters.</p>
GS-11228	<p>The name <i>name</i> has been converted to <i>other name</i> as database object name contains invalid character(s).</p> <p>The original name of the database object contains invalid characters. The database object has been renamed to a new name.</p>
GS-11270	<p>Unexpected extraction options.</p> <p>An unexpected extraction options section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</p>
GS-11271	<p>Unexpected extraction settings.</p> <p>An unexpected extraction settings section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</p>
GS-11272	<p>Unexpected generation options.</p> <p>An unexpected generation options section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</p>
GS-11273	<p>Unexpected generation settings.</p> <p>An unexpected generation settings section has been found in the 4db or 4dbx document. The schema file is corrupted and cannot be read.</p>
GS-11303	<p>The database object <i>name</i> has been newly added in schema.</p> <p>The database object has been added to the schema and is not yet found in the database. Creating a database update script will generate SQL instructions to add this object to the database.</p>
GS-11304	<p>The database object <i>name</i> has been marked as removed from schema.</p> <p>The database object has been removed from the schema and can still be found in the database. Creating a database update script will generate SQL instructions to remove this object from the database.</p>
GS-11310	Name has been modified from <i>name</i> to <i>other name</i> .

Number	Description
	The database object has been renamed in the schema and can still be found in the database. Creating a database update script will generate SQL instructions to update this object in the database.
GS-11311	Database object <i>name</i> has an invalid uuid. A new one has been generated (<i>uuid</i>). The internal identifier of the database object is invalid and has been replaced with a new one.
GS-11315	Column <i>type</i> has been changed from <i>type</i> to <i>other type</i> . The type of the column has been changed in the schema. Creating a database update script will generate SQL instructions to update the column in the database.
GS-11316	Not null property has been changed from <i>value</i> to <i>other value</i> . The value of the 'Not null' property of the column has been changed in the schema. Creating a database update script will generate SQL instructions to update the column in the database.
GS-11318	Column <i>column</i> in table <i>table</i> uses Informix specific type: <i>type</i> . The column uses an Informix specific data type.
GS-11320	Constraint or index type has been changed from <i>type</i> to <i>other type</i> . The type of the constraint has been changed in the schema. Creating a database update script will generate SQL instructions to update the constraint in the database.
GS-11322	Constraint or index columns of <i>name</i> has been changed from <i>columns</i> to <i>other columns</i> . The columns of the constraint have been changed in the schema. Creating a database update script will generate SQL instructions to update the constraint in the database.
GS-11323	Constraint or index <i>name</i> column order of <i>name</i> conflicts with column order of <i>foreign key</i> . Indicates the order of columns in the foreign key is not the same as the one defined for the constraint or index.
GS-11325	Foreign key table has been changed from <i>table</i> to <i>other table</i> . The table associated to the foreign key has been changed in the schema. Creating a database update script will generate SQL instructions to update the foreign key in the database.
GS-11326	Foreign key referenced table has been changed from <i>table</i> to <i>other table</i> . The reference table associated to the foreign key has been changed in the schema. Creating a database update script will generate SQL instructions to update the foreign key in the database.
GS-11327	Foreign key columns of <i>foreign key</i> has been changed from <i>columns</i> to <i>other columns</i> .

Number	Description
	The columns associated to the foreign key have been changed in the schema. Creating a database update script will generate SQL instructions to update the foreign key in the database.
GS-11328	<p>Foreign key referenced columns of <i>foreign key</i> has been changed from <i>columns</i> to <i>other columns</i>.</p> <p>The reference columns associated to the foreign key have been changed in the schema. Creating a database update script will generate SQL instructions to update the foreign key in the database.</p>
GS-11329	<p>Cascade delete property has been changed from <i>value</i> to <i>other value</i>.</p> <p>The value of the 'Cascade delete' property of the foreign key has been changed in the schema. Creating a database update script will generate SQL instructions to update the foreign key in the database.</p>
GS-11330	<p>Foreign key <i>name</i> column order of <i>column</i> conflicts with column order of <i>other column</i>.</p> <p>Several columns are defined in the foreign key with the same order. The order of the column is automatically updated to solve the issue.</p>
GS-11340	<p>Constraints or indexes <i>constraint</i> and <i>other constraint</i> are defined on same set of columns.</p> <p>Both constraints are defined on the same set of columns.</p>
GS-11341	<p>Foreign key <i>foreign key</i> references columns which are not part of primary key or unique constraint.</p> <p>No primary key or secondary key is defined on the columns referenced by the foreign key.</p>
GS-11342	<p>Constraint cannot be placed on a BYTE or TEXT column.</p> <p>A unique constraint has been defined on column of type BYTE or TEXT.</p>
GS-11343	<p>Non-unique index <i>index</i> conflicts with constraint <i>constraint</i>.</p> <p>A non-unique index has been defined on the same set of columns has a unique constraint.</p>
GS-11350	<p>File contains orphan property <i>property</i>, clean document settings to remove it.</p> <p>The meta-schema file was created using a set of template containing dynamic properties which are not defined in the current set of template any more. Use the appropriate action to clean up the meta-schema and to remove those unused properties.</p>
GS-11351	<p>File contains orphan property group <i>group</i>, clean document settings to remove it.</p> <p>The meta-schema file was created using a set of template containing groups of dynamic properties which are not defined in the current set of template any more.</p>

Number	Description
	Use the appropriate action to clean up the meta-schema and to remove those unused property groups.
GS-11352	<p>Node <i>node</i> contains orphan properties, clean document settings to remove them.</p> <p>The meta-schema file was created using a set of template containing dynamic properties which not defined in the current set of template any more. Some database objects are using those obsolete properties. Use the appropriate action to clean up the meta-schema and to remove those properties.</p>
GS-11360	<p>Foreign key references an inactive table.</p> <p>The table referenced by the foreign key is flagged as inactive whereas the table associated to the foreign key is active.</p>
GS-11361	<p>Foreign key table is inactive.</p> <p>The table associated to the foreign key is flagged as inactive whereas the table referenced by the foreign key is active.</p>
GS-11400	<p>Unknown tag <i>tag</i>.</p> <p>The meta-schema document contains unknown data. The schema file is corrupted and cannot be read.</p>
GS-11550	<p>Wrong line format.</p> <p>The data contained in the file being imported to a meta-schema has an unsupported format. The data cannot be imported.</p>
GS-11551	<p>Can't convert type for column <i>column</i> of table <i>table</i>.</p> <p>The type of the table column defined in the imported table is unsupported. The data cannot be imported.</p>
GS-11552	<p>Can't convert length for column <i>column</i> of table <i>table</i>.</p> <p>The length of the table column defined in the imported table is invalid. The data cannot be imported.</p>
GS-11553	<p>Can't convert order for column <i>column</i> of table <i>table</i>.</p> <p>The column has an unsupported order value. The import of the sch file cannot continue.</p>
GS-11600	<p>Compilation generated an empty sch file.</p> <p>The compilation of the schema document generated an empty sch file. This can happen when no valid table columns have been found in the document.</p>
GS-11650	<p>The extraction process task can't be generated.</p> <p>An error occurred while trying to create the database extraction process.</p>
GS-11651	Database name can't be empty.

Number	Description
	An empty database name has been found while trying to set up the database connection.
GS-11652	Schema file name can't be empty. An empty file name has been found while trying to generate the schema document.
GS-11653	Preparing schema extraction... This message indicates the schema extraction process is being set up.
GS-11654	Schema extraction started... This message indicates the schema extraction process has started.
GS-11655	Merging extracted schema with existing one... This message indicates the schema extraction process tries to merge the existing schema document with the extracted one.
GS-11656	Generating schema id... This message indicates a new schema identifier is being generated.
GS-11657	Schema extraction successful. This message indicates the schema extraction process has successfully completed.
GS-11658	Schema extraction failed. The schema extraction process has failed at some point. Refer to other messages to get additional information.
GS-11659	Schema extraction aborted. The schema extraction process has been aborted by the user.
GS-11660	Failed to move temporary schema file to destination. An error occurred while trying to move the temporary schema file to its destination.
GS-11661	Failed to create schema backup file. An error occurred while trying to create a backup of the schema file.
GS-11662	Failed to remove schema backup file. An error occurred while trying to delete a backup of the schema file.
GS-11700	The connection test process task can't be generated. An error occurred while trying to create the database connection test process.
GS-11750	Generating documentation for <i>meta-schema</i> ... This message indicates documentation for the schema is being generated.
GS-11751	Documentation generated at <i>location</i>

Number	Description
	This message indicates documentation for the schema has been generated and is saved at the given location.
GS-11850	Failed to extract database meta data. The extraction of the meta-schema has failed. Refer to the previous messages for more information.
GS-11851	Failed to connect to database. The connection to the database has failed. Ensure the appropriate connection information has been provided.
GS-11852	Cannot instantiate database driver <i>driver</i> . The database driver cannot be loaded. Check the driver name is correct and accessible.
GS-11853	Unsupported database <i>database</i> . This database engine is not supported by the extraction tool.
GS-11860	Table column <i>table.column</i> has not a valid datatype <i>type</i> - table is ignored. The data type of the table column is not supported by the extraction tool. The extraction of the table is cancelled.
GS-11861	<i>type</i> column <i>table.column</i> is ignored. The data type of the table column has been flagged to be ignored by the extraction tool. The table definition will not contain this column.
GS-11862	Foreign key <i>foreign key</i> is ignored due to missing table. The foreign key references a table that was not found during extraction. The meta-schema will not contain this foreign key.
GS-11870	Unexpected conversion parameter. The conversion method provided to the extraction tool is invalid.
GS-11871	Missing conversion method. The conversion method is required by the extraction tool.
GS-11872	Missing auto-increment indicator. The parameter required for the extraction of the table column is not found.
GS-11873	Missing charlen indicator. The parameter required for the extraction of the table column is not found.
GS-11874	Missing column type indicator. The parameter required for the extraction of the table column is not found.
GS-11875	Missing distinct types indicator.

Number	Description
	The parameter required for the extraction of the table column is not found.
GS-11876	Missing identity indicator. The parameter required for the extraction of the table column is not found.
GS-11877	Missing serial indicator. The parameter required for the extraction of the table column is not found.
GS-11880	No database connection. The connection to the database is not found.
GS-11881	Table name is required. The name of the extracted table is required by the extraction tool.
GS-11882	Column name is required. The name of the extracted table column is required by the extraction tool.
GS-11883	Database schema is required. The database schema / table owner is required by the extraction tool.
GS-11890	Missing database driver. The database driver has not been provided to the extraction tool.
GS-11891	Missing output file parameter. The meta-schema file has not been provided to the extraction tool.
GS-11899	<i>unexpected error</i> An unexpected error has been encountered. Refer to the error message for more information.

DB Explorer

The DB Explorer plug-in is a tool that allows you to view, create and modify data stored in a relational database.

To design a good report, it is important to know your data. DB Explorer provides the concept of "show data", where you can see the data stored in the rows of a database table. You can view this data from the table perspective, or from the business record perspective. By knowing your data, you are better equipped to write valid expressions and reports that your readers will understand.

When testing your report design, you may need to see how the report handles specific data values. For example, you may have a sales report where you want to highlight all sales that are above (or below) a certain value. You may need to modify your table data in order to have a row that meets the criterion. In a more extreme example, you may simply have the table schema without any data records. You may need to provide the sample data yourself.

The DB Explorer plug-in exists for these reasons, and more.

Layout of the view



Figure 205: Table data in DB Explorer

At the top of the view, an editable combobox allows you to select a previously written SQL query, edit the current SQL query, or enter in your own SQL query. Next to this field are three icons:

- The **Edit SQL query** icon opens the **Query Editor**.
- The **Execute query** icon executes the displayed query.
- The **Choose connection information** icon allows you to change your database connection.

Under the combobox, there are four icons for data modification.

- **Save** icon - Saves changes to the current row.
- **Insert** icon - Creates a new, blank row.
- **Copy** icon - Create a copy of the selected row.
- **Delete** icon - Delete the selected row.

The data itself displays in the scrollable table container at the bottom of the view.

When to use

Use DB Explorer to quickly view or make changes to your data. It is important to understand that, by using DB Explorer, you are making actual changes to the data in the tables; these changes are permanent. As such, DB Explorer is intended as a developer tool, not as a production tool.

Limitations

The following SQL commands are not supported:

- SQL commands that output information as text (for example, a command that shows a list of tables) will execute, however the text will not display.
- SQLite-specific commands (known as dot commands, such as `.show`) are not supported.

When you change or modify data using this tool, the changes are done under-the-covers by SQL statements. These SQL statements are governed by the rules of the database itself. The change needs to be valid by the rules of the database, in order for the query to run successfully.

You cannot use DB Explorer to view or edit binary data (BLOB data type). In addition, for some databases you cannot use DB Explorer to view or edit text-based large objects, such as the CLOB data type in IBM-Infomix database servers.

DB Explorer will always work with the actual database structure. You may have made changes to the tables and column definitions in the database meta-schema file (`.4db`), but if those changes are not made to the actual database, they are ignored by DB Explorer. Conversely, if changes are made to the underlying database structure (through the use of DDL or another database tool), the meta-schema file (`.4db`) must be manually updated using the **Database >> Update Schema** menu option.

Open DB Explorer

DB Explorer is one of many views available from within Genero Studio

While you can explicitly open DB Explorer, it will open automatically as needed when you follow the steps to show or edit data.

Using the Views menu

To explicitly open DB Explorer, you use the **Views** menu.

Select **Window >> Views >> DB Explorer**.

The DB Explorer view opens.

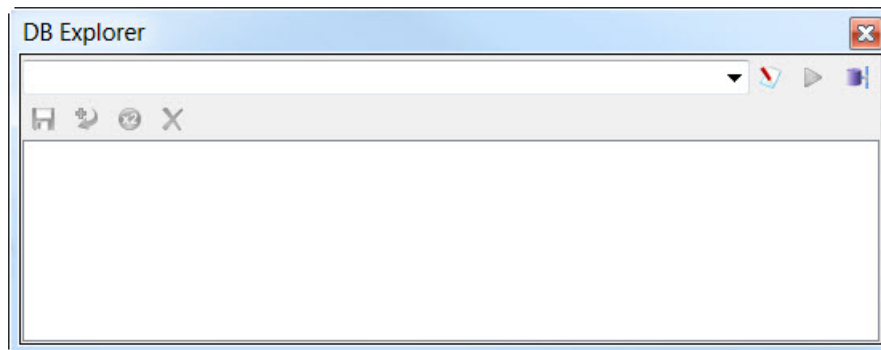


Figure 206: An empty DB Explorer view

As a next step, you will likely [specify the database connection](#).

From the meta-schema or Data Model

When you ask to show or edit data, the result set opens for you in DB Explorer. See [Show data](#) on page 334 and [Change the data](#) on page 336.

Change connection details

In order to show or edit data, you must be connected to a relational database.

1. Open the Connection information dialog.
 - From within DB Explorer view, click the **Choose Connection Information** icon.
 - With the meta-schema file (.4db, .4dbx) opened in the central work area, select **Database >> Edit Database Connection**.
 - With the meta-schema file (.4db, .4dbx) opened in the central work area, click anywhere within the white space in the meta-schema diagram and select Edit Database Connection from the contextual menu.
 - With the meta-schema file (.4db, .4dbx) opened in the central work area, in the Database Structure view, right-click on the database node and select **Edit Database Connection**.
2. On the Connection information page, complete the three areas.
 - a) Check (or de-select) the **Keep schema connection information synchronized** check box.
 When selected, the schema connection information will be permanently updated for the current meta-schema (.4db or .4dbx). This check box appears as selected and read-only when you edit the database connection for an existing meta-schema file. If you edit the connection from DB Explorer, you can uncheck this option to dissociate the DB Explorer connection information from the meta-schema connection information.
 - b) In the **Database Connection Information** section, select either **Use explicit settings** or **Use external settings**.

Use explicit settings, previous connection	You can use a previous connection that was created for the same database. The drop down list provides a list of the existing connections.
Use explicit settings, database type	You can enter the Database Type by selecting the desired type from the drop down list, and the corresponding information for that type. The Database driver for the database type is automatically entered. If other drivers exist, they are available in the drop down list.
Use external settings	Information in the <code>FGLPROFILE</code> configuration file is used to extract the corresponding connection information for the specified database. Genero Studio will use the schema name that you entered to check for any related entry in the <code>FGLPROFILE</code> configuration file, and will use those values to define the connection. See information on the <code>FGLPROFILE</code> file in the <i>BDL User Guide</i> .
 - c) In the **Database User Information** section, provide the necessary database user details. The required information varies based on the database type selected. See [Database server/user information](#) on page 308.
3. Click **Test Connection** to verify your configuration.
4. Click **OK** to close the dialog.

Show data

You can show data for a table, for a set of columns, or for a business record.

You have several choices to make, to include whether you wish to view some or all columns of data, or whether to start from the meta-schema file or from DB Explorer itself. You can also view, but not edit, the data for a business record.

You cannot use DB Explorer to view or edit binary data (BLOB data type). In addition, for some databases you cannot use DB Explorer to view or edit text-based large objects, such as the CLOB data type in IBM-Informix database servers.

Show table data (start with meta-schema diagram)

Follow these steps to view data for a specific table, using the meta-schema diagram as the starting point.

1. Open the meta-schema (.4db or .4dbx) diagram.
2. Right-click anywhere within the table object and select **Show/Edit Table Data**.
DB Explorer opens, with the table data displayed.

Show table data (start with DB Explorer)

Follow these steps to view data for a specific table, using DB Explorer as the starting point.

1. Open DB Explorer.
2. Click the **Edit SQL query** icon.
The **Query Editor** opens.
3. Select the **Edit table data** radio button.
4. From the combobox, select the table.

Tip: If you were to select Execute at this time, the table data would display in edit mode.

The SQL query used to select the rows from the selected table displays.

5. Select the **Execute query** radio button.
6. Click **Execute**.
The table data displays.

Show data for select columns

Follow these steps to view data for a subset of table columns.

1. Open the meta-schema (.4db or .4dbx) diagram.
2. Within in the table object (the box containing the table details), press the CTRL key and select one or more columns.

Tip: As another options, you can do your column selection from within the **Database Structure** view.

The column names of the selected columns turn red.

3. Right-click within the table object and select **Show Column(s) Data**.
DB Explorer opens, with the data displayed for the selected columns.

Show data for a business record

A business record can contain data from multiple tables, depending on how it was defined. Follow these steps to view the data for a business record.

1. Open the business record (.4rdj) diagram.
2. Right-click in the Business Record object and select **Show Data**.

Tip: As another options, you can also right-click on the business record node in the **Structure View** view.

Change the data

You may need to change your data, in order to test your report designs.

While you can write SQL statements, DB Explorer is designed to allow you to make data changes directly within the user interface, to the data shown.

Use DB Explorer to quickly view or make changes to your data. It is important to understand that, by using DB Explorer, you are making actual changes to the data in the tables; these changes are permanent. As such, DB Explorer is intended as a developer tool, not as a production tool.

When you change or modify data using this tool, the changes are done under-the-covers by SQL statements. These SQL statements are governed by the rules of the database itself. The change needs to be valid by the rules of the database, in order for the query to run successfully.

Show data from a table in edit mode

To edit the data without having to hand-write SQL statements, you must start with the meta-schema file.

1. Open the meta-schema (.4db) file.
2. Right-click on a table object and select **Show/Edit Table Data**.
The table data shows in DB Explorer, in edit mode.

Show data from a different table in edit mode

When in edit mode, you can switch to a different table for editing.

1. Click the **Edit SQL query** icon. The **Edit SQL query** icon consists of a pad and pencil image.
The **Query Editor** dialog opens.
2. Select the **Edit table data** radio button.
3. From the combobox, select the table.
4. Click **Execute**.
The table data shows in DB Explorer, in edit mode.

Update data

Before you begin, you must be viewing the data in edit mode.

This procedure tells you the recommended method for updating one or more values.

1. Double-click a table cell.
2. Change the value in the cell.
3. Repeat for any other fields within the same record.

Note: If you switch to a different row, DB Explorer will save the changes to the current row automatically.

4. When you have finished updating values for a record, click the **Save** icon. The **Save** icon is a picture of a computer disk.

Insert a row

Before you begin, you must be viewing the data in edit mode.

This procedure tells you the recommended method for inserting a new row.

1. Click the **Insert** icon. The **Insert** icon consists of a plus sign and an arrow.
An empty row appears at the end of the row listing, with the cursor in the first column / field.
2. Enter a value in the field, and press TAB to move to the next field. To leave the field blank, simply press TAB. Repeat until all fields are populated.

3. Click the **Save** icon. The **Save** icon is a picture of a computer disk.

Note: If you tab past the last field, DB Explorer will insert the new row for you automatically.

The row is saved. A success message is written to the status bar.

Note: If there is an error, review the Output view to identify the issue. See [Execute a query](#) on page 339.

Duplicate a row

Before you begin, you must be viewing the data in edit mode.

This procedure tells you the recommended method for creating a copy of a row. It is assumed that you will then modify one or more of the fields, before saving the copy as a new row.

1. Select a row.
2. Click the **Duplicate** icon. The **Duplicate** icon consists of "x2" inside a green circle.
3. A new row opens at the bottom of the list, with the values of the originally selected row duplicated.
4. Make modifications to the fields in the row.

Tip: Be sure to change the value of the primary key field, to avoid returning a SQL constraint error.

5. Click the **Save** icon. The **Save** icon is a picture of a computer disk.
The row is saved. A success message is written to the status bar.

Note: If there is an error, review the Output view to identify the issue. See [Execute a query](#) on page 339.

Delete a row

Before you begin, you must be viewing the data in edit mode.

This procedure tells you the recommended method for deleting a row.

1. Select a row.
2. Click the **Delete** icon. The **Delete** icon consists of a red "X".
3. A dialog appears asking you to confirm your delete request. Click **Yes** to delete the row.
The row is deleted. A success message is written to the status bar.

Limit rows

You can set a limit to the number of rows to display.

Without a limit, you could end up retrieving massive amounts of data. By default, the limit is set to 1000. You can remove all limits by entering zero (0).

When the row limit is met, a message displays in the status area of DB Explorer stating that the limit has been met.

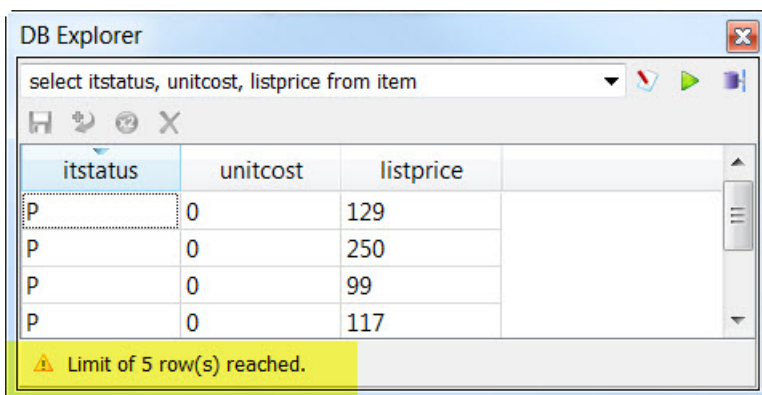


Figure 207: Message displays when limit is met

Regardless of the limit, the data returned always starts with the first row returned by the database server. There is no mechanism to change the order of the rows fetched, or which rows are returned, without altering the SQL by hand.

To set the limit on the number of rows returned:

1. Select **Tools >> Preferences**.
2. Select **Meta-Schema manager Preferences**.
3. Under **DB Explorer options**, specify the maximum number of results (or records) to display and retrieve.

Write a SQL query by hand

DB Explorer is designed to allow users to create or modify data using the graphical interface. You can, however, use DB Explorer to edit SQL queries you write by hand.

The following SQL commands are not supported:

- SQL commands that output information as text (for example, a command that shows a list of tables) will execute, however the text will not display.
- SQLite-specific commands (known as dot commands, such as `.show`) are not supported.

When you change or modify data using this tool, the changes are done under-the-covers by SQL statements. These SQL statements are governed by the rules of the database itself. The change needs to be valid by the rules of the database, in order for the query to run successfully.

You cannot use DB Explorer to view or edit binary data (BLOB data type). In addition, for some databases you cannot use DB Explorer to view or edit text-based large objects, such as the CLOB data type in IBM-Informix database servers.

Note: The Query Editor does not provide any language syntax assistance.

1. Open DB Explorer.
2. Establish a connection to a database. See [Change connection details](#) on page 333.
3. Click the **Edit SQL query** icon.
The **Edit query** dialog opens.
4. Enter your query.
You have three options:
Write and execute a new query by hand.
 - a) Enter your query in the text box.
 - b) Click **Execute**.

Edit a previously executed query.

- a) In the **Previous queries** combobox, select the query to edit.
The query appears in the text box.
- b) Edit the query.
- c) Click **Execute**.

Edit a query generated by the Edit table data wizard.

Tip: To use this method, you must have opened DB Explorer from the meta-schema diagram.
See [Show data](#) on page 334.

- a) Select the **Edit table data** radio button.
- b) From the combobox, select the table.

Tip: If you were to select Execute at this time, the table data would display in Edit mode.

The SQL query to select the rows from the selected table displays.

- c) Select the **Execute query** radio button.
- d) Edit the query.
- e) Click **Execute**.

Execute a query

Executing your query is the last step to viewing or modifying the data.

When you execute your query, it will either be successful or it will have a query execution error.

DB Explorer will always work with the actual database structure. You may have made changes to the tables and column definitions in the database meta-schema file (.4db), but if those changes are not made to the actual database, they are ignored by DB Explorer. Conversely, if changes are made to the underlying database structure (through the use of DDL or another database tool), the meta-schema file (.4db) must be manually updated using the **Database >> Update Schema** menu option.

Successful execution of the query

If the query is a SELECT statement, and it is successful, the results appear in DB Explorer. The complete SQL query is written to the Output view.

If the query is an INSERT, UPDATE or DELETE statement, and it is successful, you receive no visual confirmation. The complete SQL query, however, is written to the Output view.

Query execution error

When you have a SQL error, a message displays in a pop-up window. This message provides you with the error message being returned by the database, along with a message ID.

The error message, including the complete SQL query, are also written to the Output view.

Project Manager

Project Manager is a tool to manage the organization and build of executables from the program's source files.

From Project Manager you can easily create or import projects; add and edit source files in Code Editor; add and edit forms in Form Designer; build, link, and execute programs; execute programs with the Profiler; and debug with the Graphical Debugger.

- [Genero project file \(4pw\)](#) on page 340
- [Quick Start: Create a project](#) on page 340
- [Creating new projects](#) on page 341
- [Organizing projects](#) on page 342
- [Building and linking programs](#) on page 343
- [Packaging](#) on page 353
- [Locate a file \(starting at Project Manager\)](#) on page 353
- [Project Manager Reference](#) on page 354

Genero project file (4pw)

A Genero Project (4pw) is an XML file that manages the source files and the properties for building and executing programs. Genero Studio displays a project file in a convenient and visible way as a tree view in the Projects view.

A project includes:

- the name, type, and location of the source code and executable files
- the database meta-schema, if used
- the name, type and description of each node
- the parent-child relationships between nodes
- properties such as file paths, environment variables, and dependencies
- one or more sub-projects, if used

The project structure has no connection to the structure of the files on the disk. Files can be deleted and moved from projects without affecting the actual file on the disk.

A default project file contains nodes for an **application**, a **library**, and **databases**.

Quick Start: Create a project

Quickly create and test a new project. This example creates a project using Genero Studio sample source files.

1. Select **File >> New >> Genero Files, Simple Project**.
2. Save and name the project.
3. Expand the tree to see all default nodes in the project.
4. Right-click on the **Application** node and select **Add File**. Navigate to the `My Genero Files/samples/HelloWorld` directory and select and open `HelloSource.4gl` and `HelloSource.4fd`.
5. Add a new **Library** node. Right-click on the **Group** node and select **New Library**. Name the library `Forms`. Drag the `HelloForm.4fd` file to the **Forms** node and drop it on top of the `Forms` node.

6. To identify that the application is dependent upon the files in the new Forms library, right-click on the **Application** node and select **Advanced Properties**. Notice that the Forms node is not checked. Check the Forms node to indicate that the application is dependent on the files in this library. Select **OK**.
7. Right-click the **Group** or **Application** node, and select **Build** to compile and link the files into an executable program. Check the output of the build in the **Output** view.
8. If the build is successful, right-click the **Application** node and select **Execute** to run the program through Genero Studio. The compiled and executable files for the program are stored in the **Target Directory** specified.

Creating new projects

Information about creating new projects.

- [Create a new project](#) on page 341
- [Import existing files as a new project](#) on page 341
- [mkproject - Convert a Makefile to a project](#) on page 342
- [Connect to existing build systems](#) on page 342

Create a new project

Create a new project to manage the project source files, libraries, and database schemas.

1. Select **File>>New** from the Genero Studio main menu to create a Project (4pw).

Option	Description
Genero Files, Simple Project	Creates a simple project used for applications that are not code-generated.
Genero Files, Mobile Project	Creates a simple project used for mobile apps that are not code-generated.
Design, Managed Project	Creates a project for use with generated (BAM) applications. See What is Business Application Modeling (BAM)? on page 192
Design, BAM Mobile Project	Creates a project for use with generated (BAM) mobile applications.

2. After creating a Project (4pw), use **File>>Save** to specify the location of the file in the file system.

Import existing files as a new project

Files in a file system can be imported as a new project using the **Project >> Import** menu option.

Before you begin, [Set up a remote environment](#) if your source files are on a remote host.

1. If the source files to be imported contain reference fields from a database table, create or import a meta-schema file (4db) for the database, if not already completed.
2. [Create a new, simple project](#) to manage the files using Project Manager.
3. Use the [Import Project](#) from the Projects menu to import the directory as a new project. To make sure that only source files appear in the project, either clean all binaries and generated files from the directory before importing or set the import filters in the [Import Project](#) dialog to include/exclude files of various types.

Genero Studio will organize the files into the project. Each 4g1 file with a MAIN will reside in an Application node. All other files will be located in Library nodes. You can organize the files how you wish.

4. If sources contain files that are preprocessed to generate other files, set up the appropriate [build rules](#). (For example, if a file with an extension of 5g1 generates a 4gl and then builds it: Add a build rule 5g1 # 42m.)

5. Once the source files are imported into the project, if the **Compute dependencies** checkbox is selected, the import process will automatically fill the required information for any libraries that must be linked in the same project. Right-click the **Application** node and select **Advanced Properties, Dependencies** to check the appropriate dependencies for each application or library.
6. [Add the meta-schema file](#) to the **Databases** node of your project.
7. Set the other [properties](#) for the nodes in the project, as needed. For example:
 - a) To change the default location of the output files when the program is built, set the [Target Directory](#) property for the Group or Application nodes. The default target directory is **\$ProjectDir/bin**.
 - b) If there are external libraries (outside the Project) that should be linked when the application is built, add the library to the value of the [external dependencies](#) property for the application node.
8. Build the Project to compile and link the files, checking for any dependency errors.

mkproject - Convert a Makefile to a project

mkproject can be used to automatically create 4pw project files from the older build system make.

See `GSTDIR/tools/mkproject/readme.txt` on how to use mkproject.

Connect to existing build systems

To connect to an existing build system, edit the build rules or link rules to execute the building system commands.

For example, if the existing build system is `make`, you can call `make` in the link rule to create a library.

Organizing projects

Information about organizing projects.

- [Groups, Applications, and Libraries](#) on page 342
- [Using external libraries](#) on page 343
- [Setting external dependencies](#) on page 343

For information about source code management for your project, see [Source Code Management - SVN](#) on page 529.

Groups, Applications, and Libraries

Projects are organized into Group, Application, and Library nodes. The Projects view visually displays a project file (4pw) for easy management of project source files.

Group

Group nodes organize the Application and Library nodes that make up the project. Rapidly define default properties by setting them at the group level (TargetDir, Language, Compiler options, etc.) Properties defined for the Group node are inherited by all child nodes in the group.

Application

The Application node is used to generate an executable program (42r). Application nodes can contain both files and virtual folders.

Only *one* of the files in the Application node may have a `MAIN` statement: One Application node equals one executable. The name of the Application node is used as the name of the 42r file, so it must be unique and can only contain characters allowed by the file system.

The default application is shown in boldface. Use the Projects view integrated Toolbar to set a different application as the default. The options on the Build menu execute for the default application.

Library

A library node is used to group binary files into a single library and generate a library file (42x). It can contain both files and virtual folders. The name of the library node is used as the name of the 42x file, so it must be unique and can only contain characters allowed by the file system.

Libraries should be used when creating a set of features having a common goal, like the logic of an application, a library of mathematical functions, etc.

A library can also be used to group other project files together (images, styles or other resources).

If a library node contains no 4gl file, no 42x is built.

A library from a different project can be added to a project using the right-click menu option **Add External Project**.

Important: A library must be linked to any application in which it will be used by right-clicking the application node and selecting [Advanced Properties, dependencies](#). The checkbox for any required library must be checked.

Using external libraries

Libraries defined in one project can be used in other projects.

To create a link to external libraries, right-click on a Group node in the projects view and select Add External Project. A new node is created with the name of the external 4pw file.

When an external project is added, all of its projects, application and libraries are automatically added as children. The child nodes are in grey because they are in *read-only* mode; they cannot be moved, renamed, and so on. Their properties are also *read-only*.

It is important to set the dependencies for any application or standard library node that requires the use of this library, using **Advanced Properties**. The new node, with all its libraries as children, will be shown in the Dependencies property page; check each library that should be linked with the application or standard library.

Setting external dependencies

External dependencies are files that are not a part of the project, but should be included in the linking process.

External libraries are mainly used when a library is used in many projects and is not intended to be modified, or if third party libraries are used for which the sources are not available.

Select the Application or Library node and in the **Project** view, set the **External dependencies** property to the list of files that should be included in the linking process.

The directories containing these files must be added to the FGLLDPATH environment variable. See [Genero variables](#).

Building and linking programs

Build rules compile each file in a project. Link rules create the applications and libraries. Execution rules execute the application. Build rules, link rules, and execution rules together make up a Language in Genero Studio.

Warning: A rebuild of a project may not automatically be done when files within the project are updated. It is the responsibility of the developer to recompile the appropriate parts of the project.

- [What are build rules](#) on page 344
- [Add/Edit a build rule](#) on page 345
- [Example: How build rules work](#) on page 346
- [Languages](#) on page 344

- [Link rules](#) on page 347
- [Execution rules](#) on page 347
- [Command line options for build, link, execution rules](#) on page 347
- [Environment variables](#) on page 348
- [Pre/Post compile](#) on page 351
- [Pre/Post link](#) on page 351
- [gsmake - Command line option to build projects](#) on page 352

Languages

A language is a named set of build rules (to compile files), link rules (to link application or library nodes), execution rules (to execute/debug/profile an application) and a set of environment variables.

Genero Studio comes preconfigured for some languages. You can add your own languages.

You select which language to use by setting the **Language** property on an application, group or library node.

What are build rules

Build rules are used to compile each file in a project.

Global and Project build rules can be modified. New build rules can be created.

Build rules are grouped into the following hierarchy:

- Project - If relevant build rule found under Project, it is used.
- Specific - If no relevant build rule is found under Project, then relevant build rule under Specific is used.
- Global - if no relevant build rule is found under Project or Specific, then relevant build rule under Global is used.
- Default - if no relevant build rule is found under Project, Specific, or Global, then relevant Default build rule is used.

Only one build rule can be active for a specific file type.

Access the Build Rules dialog using **Tools >> Global setup >> Edit Build Rules**, **Tools >> Specific setup >> Edit Build Rules** or right-click on the **Project** view and select **Edit Build Rules**.

Add/Edit a build rule

From the Build rules dialog you can add or edit a build rule.

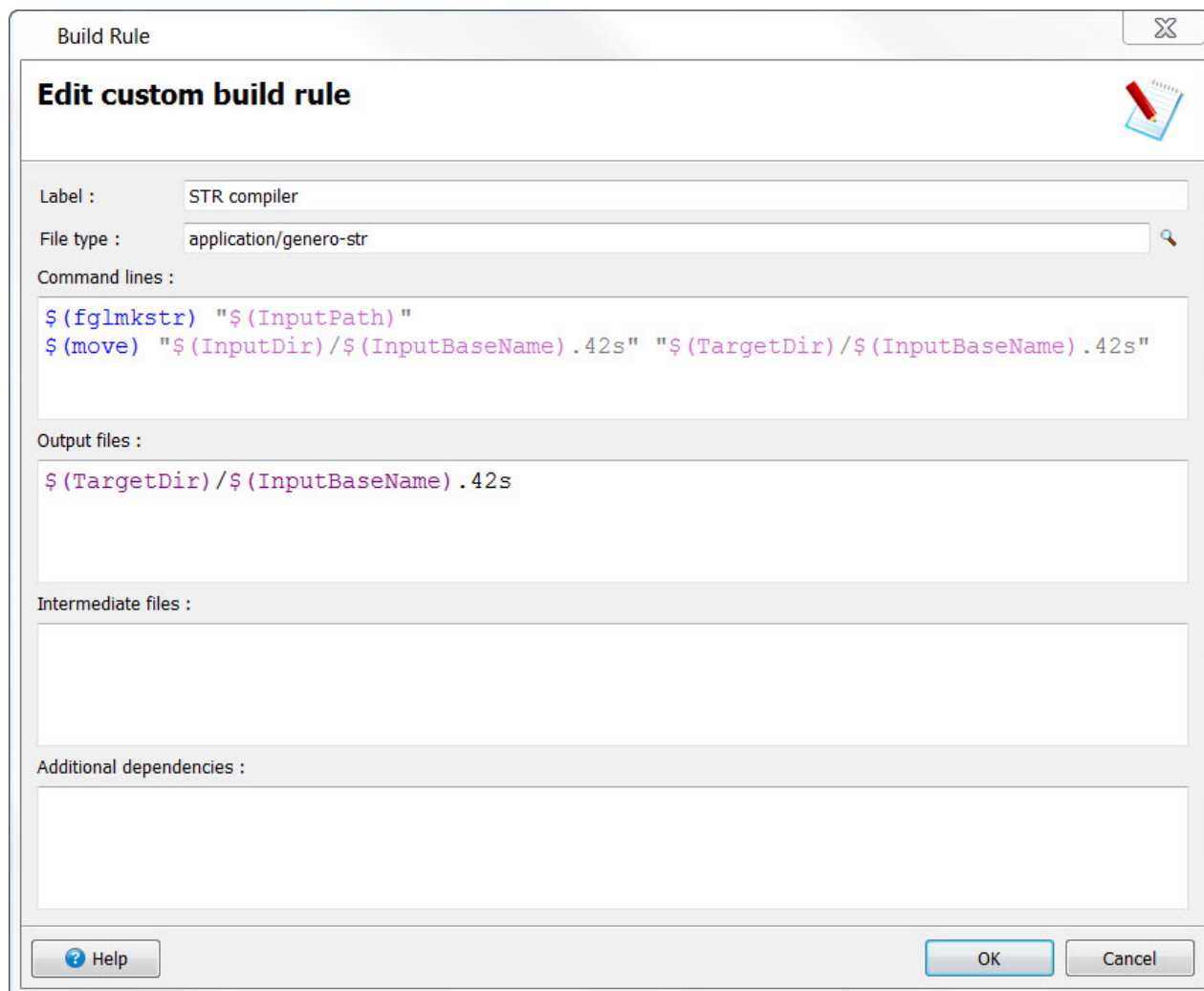


Figure 208: Add/Edit Build Rule dialog

Label

Enter a label to be assigned to the rule.

File type

Defines the MIME-type of the source files that can be compiled with this Build rule; click the icon on the right to display the File Type Selection dialog. Select the file type from a list generated from the [File Associations](#) in **Genero Studio Preferences**. Use the Search field to limit the display of the list; enter the file extension, or the words Genero or studio, for example, to display only the corresponding file types. The icons can be used to display the list by category, and to expand or contract a category.

Output files

Enter the list of files that are generated by the Build rule. All the [predefined file node variables](#) can be used when specifying the path of the file.

Note: When entering file paths, do not use quotations surrounding the file path. That is only necessary in the **Command line** field.

Intermediate files

Indicates which files are generated during compilation, as intermediate files, before generating the output file. This displays the Intermediate files in the Project view.

Note: When entering file paths, do not use quotations surrounding the file path. That is only necessary in the **Command line** field.

Additional dependencies

The additional dependencies is a list of files used to verify that the build is up to date. If one of these files has a modified date more recent than any of the output files, the build is considered not up to date and the build rule will be executed.

Note: When entering file paths, do not use quotations surrounding the file path. That is only necessary in the **Command line** field.

Example

For a preprocessor named `mypp` for source files (`*.my`), the Build Rule for the mime type "**text/my**" would be:

```
mypp $(InputPath) -o $(TargetDir)/$(InputBaseName).4gl
$(Fglcomp) $(TargetDir)/$(InputBaseName).4gl
$(delete) $(TargetDir)/$(InputBaseName).4gl
```

Redirect output

When a command is executed in a build, link, execution rule or in a user action, the output (standard or error) is displayed in the Output view. You can redirect the output to a file. For example, if you write `echo "Hello World"` in a build rule, Hello World is displayed in the Output view. To redirect the output to a file, use standard Linux™ syntax such as: `echo "Hello World" > "c:\file.txt"`. Supported syntax is `>`, `1>`, `>>`, `2>`, `1>&2`, `2>&1`.

Example: How build rules work

Example: myfile.4gl

The process used by Project Manager to compile a file called **myfile.4gl** is:

1. Project Manager first searches the language to be used as defined in the parent application or library node.
2. Project Manager determines the MIME type of this file; it is "application/genero-4gl".
3. Then, Project Manager looks for a corresponding build rule in the Project build rules. After searching the Project build rules, if no rule is found that handles the "application/genero-4gl" MIME type, it searches the Template build rules, and then the Global build rules and then the Default build rules.
4. Finally, Project Manager executes the commands defined in the build rules after having replaced the variables.
5. If no build rule is found, the file is skipped and the next one is processed.

Link rules

Link rules create the applications and libraries.

A link rule executes when you build an application or library node.

Execution rules

An execution rule is executed when you run, debug, or profile an application node.

Command line options for build, link, execution rules

Special command line options can be used for build, link, and execution rules.

Table 95: Command line operands

Command line / syntax	Description
\$(move) \$(move) sourceFilePath destinationFilePath	Moves the given file or directory to the given destination in a platform independent way.
\$(copy) \$(copy) sourceFilePath destinationFilePath	Copies the given file or directory to the given destination in a platform independent way.
\$(delete) \$(delete) filePath1 filePath2 ...	Removes the given files or directories in a platform independent way.
\$(4dbcomp) \$(4dbcomp) sourceFile[4db]	Builds the schema file (sch) from the database file (4db).
\$(4fdcomp) \$(4fdcomp) sourceFile[4fd]	Builds the compiled form file (42f) from the form file (4fd).
\$(percomp) \$(percomp) [options] sourceFile[per]	The fglform tool compiles form specification files into XML formatted files used by the programs.
\$(fglcomp) \$(fglcomp) [options] sourceFile[4gl]	The fglcomp tool compiles BDL program sources files into a p-code version.
\$(fglkmmsg) \$(fglkmmsg) [options] sourceFile[.msg] [outFile.iem]	The fglkmmsg tool compiles message files into a binary version used by the BDL programs.
\$(fglkmstr) \$(fglkmstr) [options] sourceFile[.str]	The fglkmstr tool compiles localized string files.

Command line / syntax	Description
<pre>\$(fglwsdl)</pre> <pre>\$(fglwsdl) [options] <filename url></pre>	<p>Calls the fglwsdl tool for creating a web services program.</p>
<pre>\$(generate)</pre> <pre>\$(generate) [options] filename</pre>	<p>The \$(generate) command creates an intermediary XML file from modeled entities.</p>
<p>tclsh on page 269</p> <pre>tclsh [options] filename.xml</pre>	<p>The tclsh executable generates the final file by using both a Tcl template file and the intermediary XML file created by the \$(generate) command.</p>
<p>\$(tcl) - deprecated on page 270</p> <pre>\$(fglrun)</pre> <pre>\$(fglrun) [options] program</pre>	<p>Calls fglrun tool, the runtime system program that executes p-code programs.</p>
<pre>\$(blockpoint)</pre> <pre>\$(blockpoint) [options] filename</pre>	<p>The \$(blockpoint) command manages user added code by extracting or injecting code between BLOCK and POINT tags in a generated 4gl file.</p>
<pre>\$(gstdebug)</pre> <pre>\$(gstdebug)</pre>	<p>Calls internal Genero Studio debugger.</p> <p>Execution rules only.</p>
<pre>\$(gstrun)</pre> <pre>\$(gstrun)</pre>	<p>Calls command to run an application through Genero Studio.</p> <p>Execution rules only.</p>

Environment variables

Information about environment variables.

- [Add or edit environment variables](#) on page 144
- [What determines the value of an environment variable](#) on page 350

Add or edit environment variables

The **Environment Variable** dialog is used to add and edit environment variables.

When the **Environment Variable** dialog appears, enter:

Type	The type of environment variable. Options are Value, Value List, Directory, Directory List, File, or File List.
Name	The name of the environment variable.
Value	The value of the environment variable. When entering the value, if the type is Value List, Directory, Directory List, File, or File List, select the ellipses (...) to browse for the correct value.

If the value contains a variable name, that name must be prefaced with \$ and enclosed in parenthesis; for example `$(FGLLDPATH)`.

The list separator is always a semicolon (;) on all systems (Windows™ and UNIX™). The directory separator in a path is always a slash (/) on all systems.

Tip: Use the semicolon to separate directories in a list, and the slash (/) as the separator in a path, for portability of projects across operating systems.

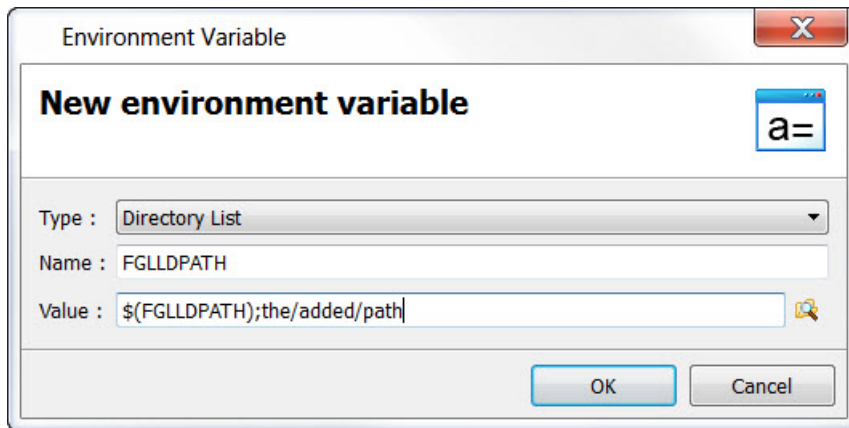


Figure 209: Setting FGLLDPATH

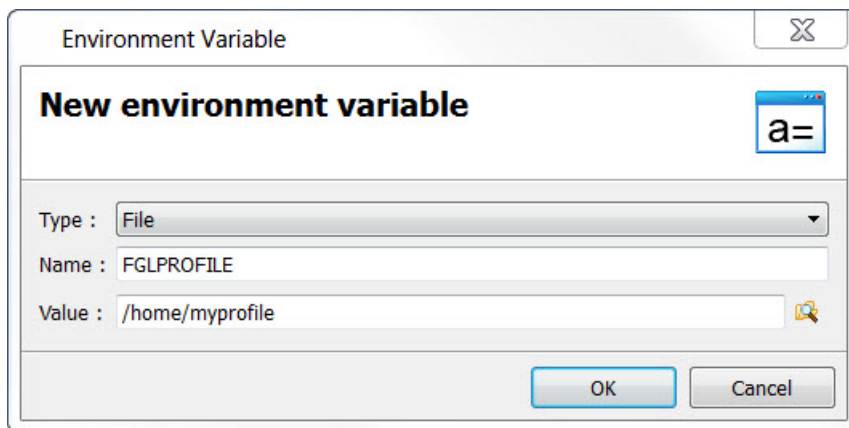


Figure 210: Setting FGLPROFILE

Reusing existing environment variables

A variable defined for a parent or ancestor node can also be reused in definitions for a child node:

For example:

- The parent node defines: `MY_VALUE=hello`
- The child node can reuse the parent node variable: `MY_COMPLETE_VALUE=$(MY_VALUE) world`
- The final value of `MY_COMPLETE_VALUE` is "hello world".

For example:

- The parent node defines: `MY_VALUE=foo`

- The child node can reuse the parent node value and redefine the variable: `MY_VALUE=$(MY_VALUE) bar`
- The final value of `MY_VALUE` will be "foo bar".

As a result, the System environment variables or Genero Studio Configuration variables can be reused in User Variable definitions within Project Manager.

What determines the value of an environment variable

Environment variables define the environment used by the compiler and executables launched from Project Manager. Where and when the environment variable is important.

Order of precedence

An environment variable can be set in multiple places, inside and outside of Genero Studio. Environment variables have an order of priority (high to low):

1. The environment defined on the node itself (Genero variables and User variables). This has the highest priority.
2. The environment of the node's parents (ancestors).
3. The environment of the node's dependencies.
4. Environment variables defined in the current [language](#).
5. Environment variables defined in [Environment sets](#) on page 140.
6. System environment variables. This has the lowest priority.

The order of priority allows you to override both system environment variables (6, above) or environment variables defined in a Genero Studio configuration (5, above) by setting them in Project Manager (1 through 4, above).

Order within an Environment Set

Within an Environment Set, Arrow keys can be used to change the order of the variables defined. This affects the way the variables are interpreted when the program is executed. For example:

Table 96: Variable interpretation examples

Project level	Variables definition and order	Result when application is executed
Project	<code>V1="hello"</code>	
Application	<code>V2="\$ (V1) world"</code> <code>V1="goodbye"</code>	<code>V1="goodbye"</code> <code>V2="hello world"</code>
Application	<code>V1="goodbye"</code> <code>V2="\$ (V1) world"</code>	<code>V1="goodbye"</code> <code>V2="goodbye world"</code>

Note: If a foreign language [Language support \(text encoding\)](#) on page 163 is selected in Genero Studio preferences, the LANG variable must be set appropriately to correspond to the selected encoding. Set the variable for a specific [environment set](#) in Genero Studio configurations (**Tools>>Configurations**).

Example: Priority and Environment Variables

In this example, the program node in the project has a library node as a dependency. The environments have been defined as shown:

Table 97: Environment Set for levels used in the example

Level	Environment Set
System environment	<pre>PATH=/bin;/usr/bin VAR1=hello VAR2=bonjour VAR3=guten tag</pre>
My Library Environment	<pre>PATH=\$(PATH);\$(ProjectDir)/ scripts VAR1=goodbye</pre>
My Program Environment	<pre>PATH=\$(PATH);\$(ProjectDir)/ scripts VAR1=\$(VAR1) world VAR2=\$(VAR2) \$(VAR1)</pre>

If the project directory `$(ProjectDir)` is set to `"/home/joe/project"`, the environment for the program will be:

```
PATH=/bin;/usr/bin;/home/joe/project/scripts
VAR1=goodbye world
VAR2=bonjour goodbye world
VAR3=guten tag
```

Pre/Post compile

Pre-compile commands are executed just before the compilation of a file, while post-compile commands are executed just after the compilation of a file.

To set pre- or post-compile commands, right-click on a file in the Projects view and select **Advanced Properties**. The **Pre/Post Compile command** dialog opens.

Pre/Post link

Pre-link commands are executed just before the link of the application or library node, while post-link commands are executed just after the link of the application or library node.

To set pre- or post-link commands, right-click on the application or library node in the Projects view and select **Advanced Properties**. The **Pre/Post Compile command** dialog opens, select the **Pre/Post Link command** option in the **Pages** listing.

gsmake - Command line option to build projects

The groups in a project (4pw files) can also be compiled from the operating system command line, using the tool `gsmake`, which is located in the `GSTDIR/bin` directory.

Syntax

```
gsmake [options] <file_list>
```

where `file_list` is a list of target Project (4pw) files, with or without the extension.

Table 98: gsmake arguments

Argument	Description
-h	Displays help information.
-v	Displays this program name and version.

Table 99: gsmake targets with parameters

Targets	Parameter	Description
-active		Targets the application set as default in the project.
-all		Targets the complete project (default behavior).
-t	<i>TARGET</i>	Adds <i>TARGET</i> to the list of targets to build. This argument can be used multiple times to build several targets

Table 100: gsmake operations

Operations	Description
-b	Builds the target (default behavior). The files that are not up-to-date are compiled, the others are not changed.
-r	Rebuilds the target. The output files are deleted, then all files are compiled.
-c	Cleans the target. The output files are deleted.
-force-build	Forces the build/rebuild of the target. The files are compiled, whether or not they are up-to-date.

Table 101: gsmake options with parameters

Options	Parameter	Description
-j	<i>NB</i>	Sets the number of parallel jobs to <i>NB</i> (default: 1). Set <i>NB</i> to 0 to use the local computer's number of CPUs. When using this option, <code>gsmake</code> will try to start multiple compilations in parallel when

Options	Parameter	Description
		possible. This should speed up the global compilation time.
-encoding	<i>ENCODING</i>	Sets the encoding to <i>ENCODING</i> . (default: System encoding)
-disable-dependencies		Disables the computation of the dependencies database. (default: false)
-ag-GSTSETUPDIR	<i>DIRECTORY</i>	Sets the Application Generator template directory to <i>DIRECTORY</i> . Enter an absolute path, or a path relative to the <i>GSTDIR/bin/src/ag/tpl</i> directory. The default value, "default" corresponds to the multiple-dialog template directory.
-max-errors	<i>NB</i>	Sets the maximum number of erroneous files to <i>NB</i> (default: 5). Set the value of NB to 0 for an unlimited number of error files.
-generate-4pwdb		Generate the 4pwdb file only (default is False).
-wcDir	<i>WEBCOMPONENTS_DIR</i>	Web Components Directory path

Packaging

Package nodes are used to package an app for deployment to a mobile device or for distribution to users.

See [Packaging, deploying, and distributing apps](#) on page 993.

Locate a file (starting at Project Manager)

From the **Projects** view, you can locate the file in the File Browser, in the System File Browser, or in a BA diagram.

Before you begin, the **Projects** view is open.

You have a file visible in the **Projects** view that you wish to locate in an alternate view, diagram, or in the System File Browser.

1. Right-click on the file.
The contextual menu displays.
2. Select the appropriate menu option:
 - To open in File Browser, select **Locate in System File Browser**.
 - To open in the System File Browser, select **Locate in System File Browser**.
 - To open in the BA diagram, select **Locate in BA Diagram**. This option is only available for files created using the Business Application Modeler (BAM).

The desired view, diagram, or dialog opens in the selected option, showing the location of the file.

Project Manager Reference

Reference information for Project Manager.

- [Project Manager context menu](#) on page 354
- [Dialogs](#) on page 355
- [Views](#) on page 390
- [Predefined node variables](#) on page 366

Project Manager context menu

Select a node in the project and right-click to display a menu of context relevant actions. Select multiple nodes using Ctrl-click.

Table 102: Project Manager Context Menu

Menu Option	Usage
New Group	Create a new node in the project .
Import Project	Import one or more existing files to project . See Import existing files as a new project on page 341
Compute Dependencies	Remove and recompute all the dependencies between all applications and libraries.
Build	Build default application. Compile and link files in the default application.
Rebuild	Rebuild selected or default application.
Clean	Clean all of selected. Erase all output files defined in the Build and Link rules.
Open Dependency Diagram	Opens Dependency Diagram
Cut/Copy/Paste	Cut, copy, or paste from clipboard.
Rename	Rename node.
Delete	Delete selected item.
Delete from Disk	Delete file from disk.
SCM	If a file is under Version Control , additional options from the context menu are available to commit, update and revert. See Source Code Management - SVN on page 529.
New Group/Application/Library/Virtual Folder/File	Create a new node in the project .
Add External Project	Add a library from a different project to the current project. See Using external libraries on page 343.
Edit Build Rules	See What are build rules on page 344
Display Environment	Displays the values of system environment and current Genero Studio configuration environment variables and of the local node variables used by Project Manager.
Advanced Properties	Set dependencies between applications and libraries, pre/post-link commands and environment variables. See Advanced Properties dialog on page 355. The

Menu Option	Usage
Set as Default Application	Dependency property always must be set if the project consists of both application and library nodes. Set the selected application as the default. Only one application can be the default at one time. By default, the first application created is the default application. The default application node is boldfaced in the project tree. The options on the Build menu execute for the default application.
New File	Create a new file to add to an Application or Library node. See File >> New on page 94
Add Files	Locate in file system and add existing files to an application or library node. This adds a link to the given file in the project ; it does not physically move the files. All the files must be located on the same drive.
Open Dependency Diagram	See Dependency Diagrams on page 403
Add Web Service	See Add Web Service on page 910
Execute	Execute selected application. If multiple applications are selected, they will be run sequentially.
Execute with Profiler	Execute selected application with Profiler. See Profiler on page 510
Debug	Launch the Debugger for selected application. Multiple selections can not be debugged.
Open	Open the selected file in Code Editor or Form Designer, depending on the file type.
Locate in File Browser	Locate file in File Browser.
Compile File	Compile the selected file.

Dialogs

Information about Project Manager dialogs.

- [Advanced Properties dialog](#) on page 355
- [Import Project dialog](#) on page 357
- [Build Rules Configuration dialog \(Languages\)](#) on page 358

Advanced Properties dialog

The **Advanced Properties** dialog provides access to setting **Dependencies**, **Pre/Post link commands**, and **Environment variables** for Group, Library, Application, or File nodes in the project.

To access the **Advanced Properties** dialog right-click on a selected **Group**, **Library**, **Application**, or **File** node and select **Advanced Properties**.

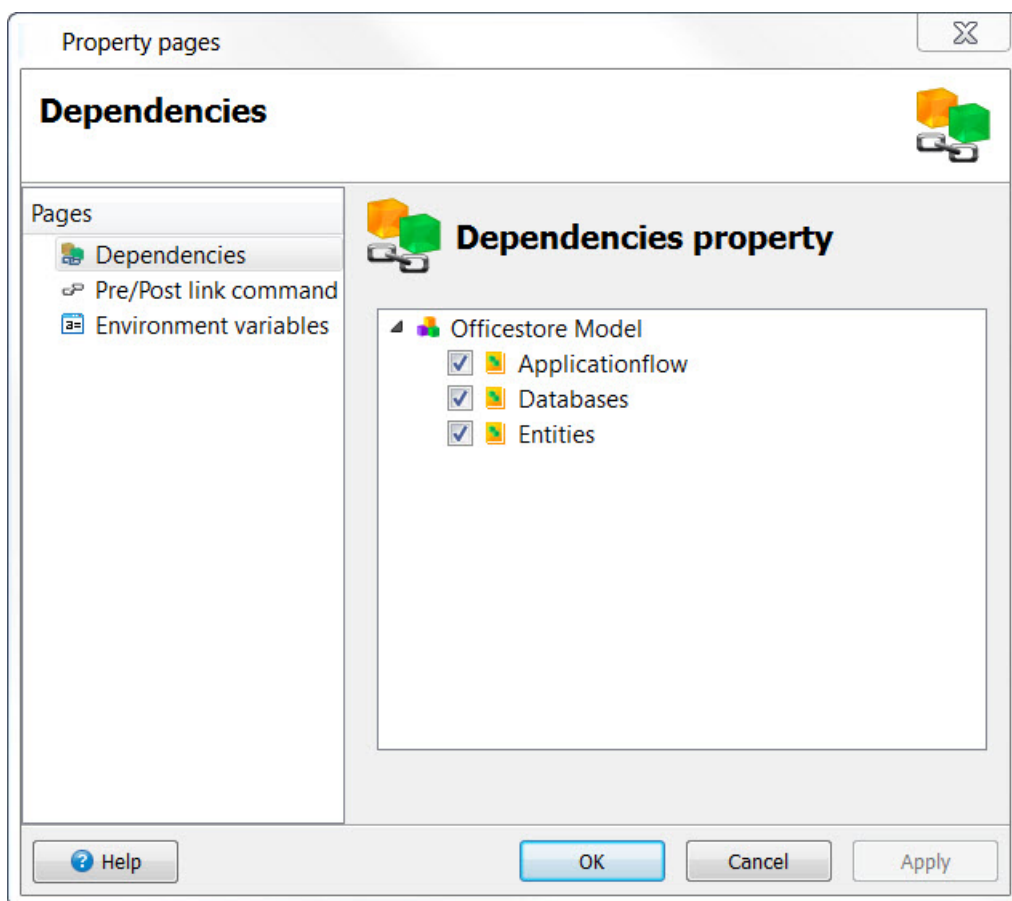


Figure 211: Advanced Properties dialog

Table 103: Advanced Properties Pages

Page	Usage
Dependencies (for Application, Library nodes)	A list of the available libraries from the current Project is displayed. Any library that must be included in the build of the node should be checked.
Pre/Post link command (for Application, Library nodes)	Shell scripts or other programs to be executed before/after linking. Enter the command in the appropriate text box. Enter the complete path of the script or program, if necessary. For example, if the script myscript is in /home/user/scripts , and this directory is included in the PATH environment variable of the system, enter myscript . Otherwise, enter /home/user/scripts/myscript .
Pre/post compile command (for File nodes)	Shell scripts or other programs to be executed before/after compiling. Enter the command in the appropriate text box, providing the complete path to the script or program if the path is not part of the PATH environment variable of the system.
Environment variables (for Group, Application, Library nodes)	Existing variables can be defined or redefined. The Genero Variables list displays a list of environment variable settings that are automatically computed by Genero Studio, and cannot be altered, although

Page	Usage
	<p>they can be redefined in Project variables. For example, if there are several applications that use different FGLPROFILE files, set a specific FGLPROFILE environment variable for each one. Predefined node variables can be used in the value. The User Variables list are variables defined by the user. Use the integrated Toolbar to add, edit, or delete a variable.</p>

Import Project dialog

Use the Import Project dialog to specify preferences when importing files from a file system into a project.

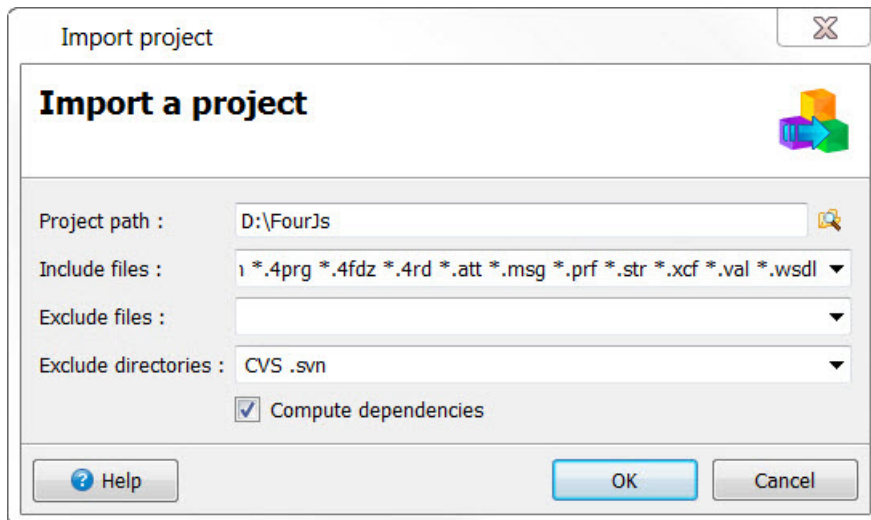


Figure 212: Import Project dialog

Project path	Enter the path and name of the directory to be imported, or use the browse button to select it from the file system.
Include files	A complete list of the files that will be imported (for example, *.4gl indicates all files with the extension 4gl). By default, files having the Genero and Genero Studio file extensions are listed; modify this list by adding or removing entries. The entries in the lists are separated by spaces.
Exclude files	Enter the files that are to be excluded from the project. Use the * symbol to indicate all files; for example, *.abc would exclude all files having the extension abc. The entries are separated by spaces.
Exclude directories	A list of excluded directories. By default the CVS directory is excluded. The entries are separated by spaces.
Compute dependencies	If checked, Project Manager will try to compute the dependencies between Application/Libraries when importing the project.

Example

In this example, all files with a `.4gl` extension in the specified path will be included, except for `testform.4gl`. The directories `CVS` and `mydir` will also be excluded.

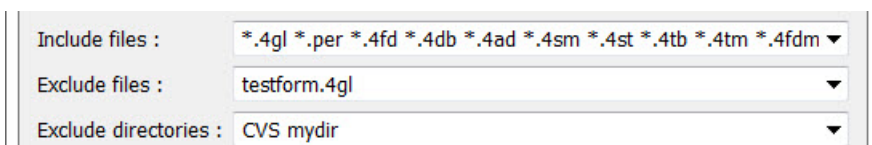


Figure 213: Import Project example

Default organization of imported files

When files are imported into Project Manager, a new **Group** node is created and listed in the Projects view. For each folder, the process is:

- A **File** node is created for each file.
- The file nodes are then added to a **Library**; the name of the folder is used as the name of the **Library** node.
- If a file contains a MAIN program block, an **Application** node is created and the **File** node is added.
- A **Group** node is then created, and all **Library** and **Application** nodes, and their nodes, are added to this **Group**.

Build Rules Configuration dialog (Languages)

The **Build Rules Configuration** dialog is used to edit a language (add, modify, delete build/link/execution rules and variables) and to add a new language or remove an existing language.

Add / edit a language

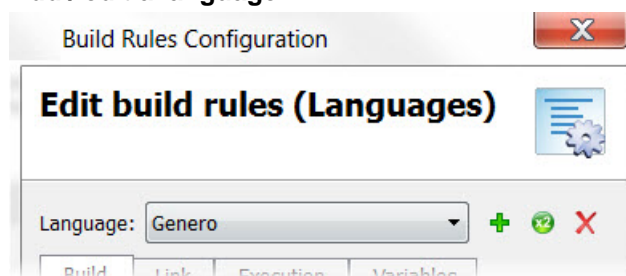


Figure 214: Build Rules Configuration dialog - Genero Studio

The integrated Toolbar allows for adding, duplicating, and deleting languages.

Add a language

Select this icon to add a new, empty language.

Duplicate the current language

Select an existing language and click this button to duplicate it.

Delete a language

Select an existing language and click this button to remove it. This action deletes only the rules and variables of the current editable level, for instance if you are editing the language from Project Manager, it will delete only the project rules and variables. If the language is empty it will remove the language.

Build tab

The Build tab in the **Build Rules Configuration** dialog is used to add, edit, and delete build rules.

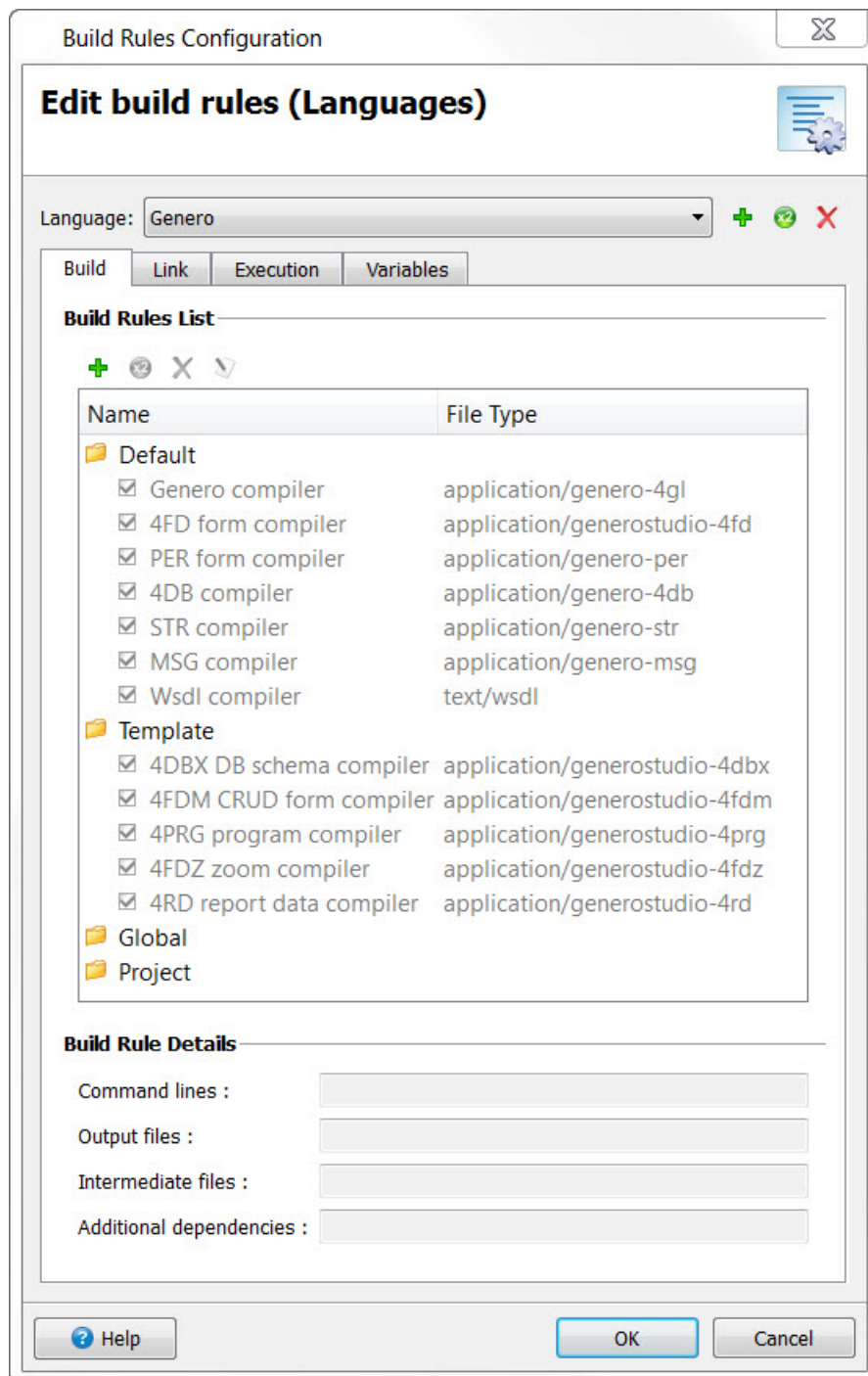


Figure 215: Build tab

Build Rules List

The Build Rules List includes the build rules [Default](#), [Template](#), [Global](#), and [Project](#) build rules for the selected language. The integrated Toolbar allows for adding, duplicating, deleting and editing build rules.

Add a build rule

Select the **Global** category (**Preferences only**) or **Project** category (**Project Manager only**) and click

this icon to add a custom rule. See [Add/Edit a build rule](#) on page 345.

Duplicate selected build rule

Select an existing rule and click this icon; the Build rule will be added to the appropriate category. Select the newly added rule and click the **Edit** icon to modify the duplicated rule.

Delete a selected build rule

Select an existing rule and click this icon to remove it.

Edit a selected build rule

Select a custom Build rule and click this icon to modify the Build Rule fields. See [Add/Edit a build rule](#) on page 345.

Link tab

The Link tab in the **Build Rules Configuration** dialog is used to add, edit, and delete link rules. A link rule is executed when you build an application or library node.

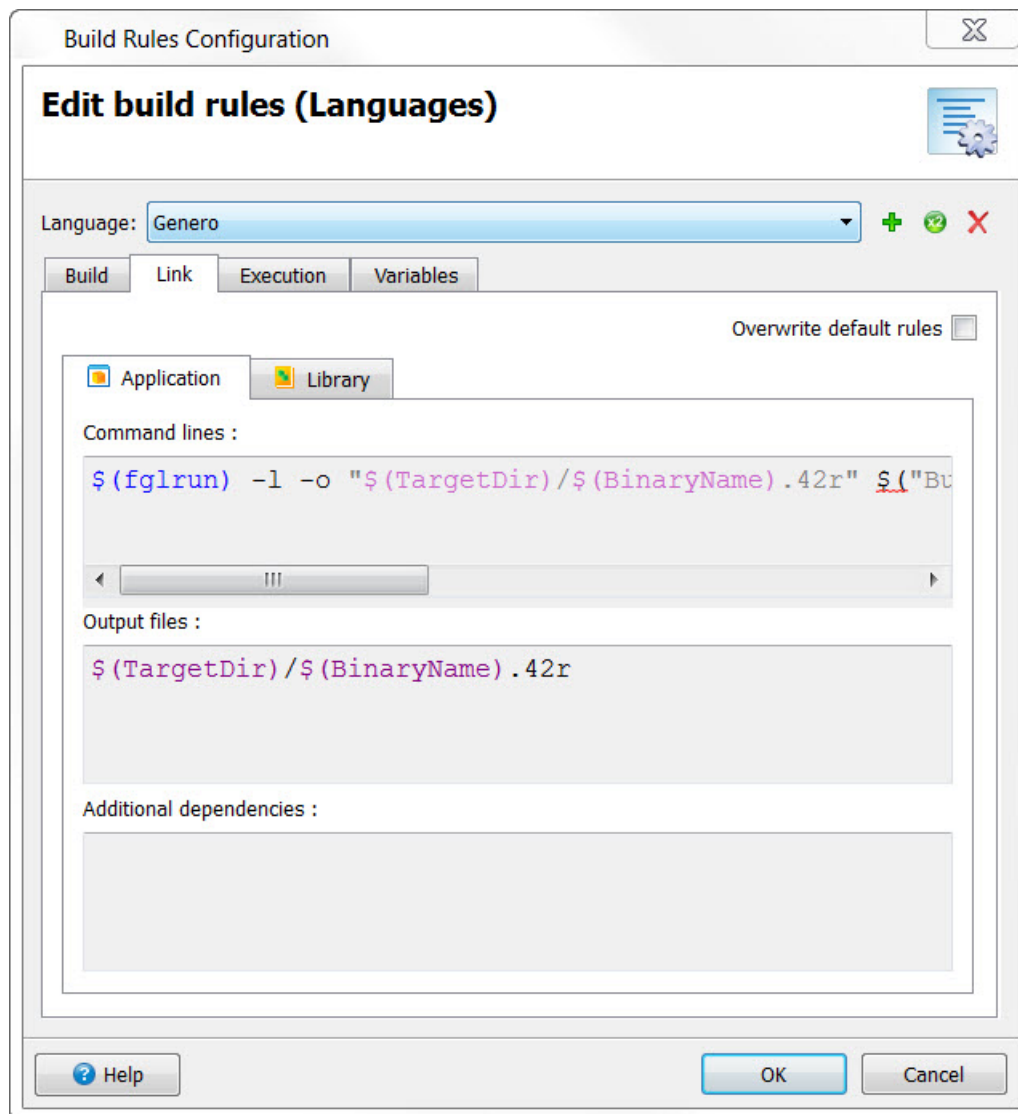


Figure 216: Link tab

Command lines

Commands that will be executed during the linking of an application / library node. See [Command line options for build, link, execution rules](#) on page 347

Output files

List of files generated by the link command. See [Predefined node variables](#) on page 366

Additional dependencies

List of files used by the link command to generate output files. If all of the listed files are less recent than the output files, the link is up to date. If the dependencies list is empty, the link rule is executed every time. If the dependencies contain one or several variables which are all empty (for example, [\\$\(BuildOutputFilePaths\)](#)), the link rule is never executed.

Overwrite default rules

If a link rule is already defined at an upper level (Default, Global, Template) the link rule is not editable unless you check the **Overwrite default rules** box to define a new link rule for the current level (Global or Project).

Execution tab

The Execution tab in the **Build Rules Configuration** dialog is used to set execution rules. An execution rule is executed when you run, debug, or profile an application node.



Figure 217: Execution tab

Run command

Command used when user runs an application.

Debug command

Command used when user debugs an application.

Profile command

Command used when user profiles an application.

Overwrite default rule

If a execution rule is already defined at an upper level (Default, Global, Template), the execution rule is not editable unless you check the **Overwrite default rules** box to define a new link rule for the current level (Global or Project).

Variables tab

The Variables tab in the **Build Rules Configuration** dialog is used to set environment variables.

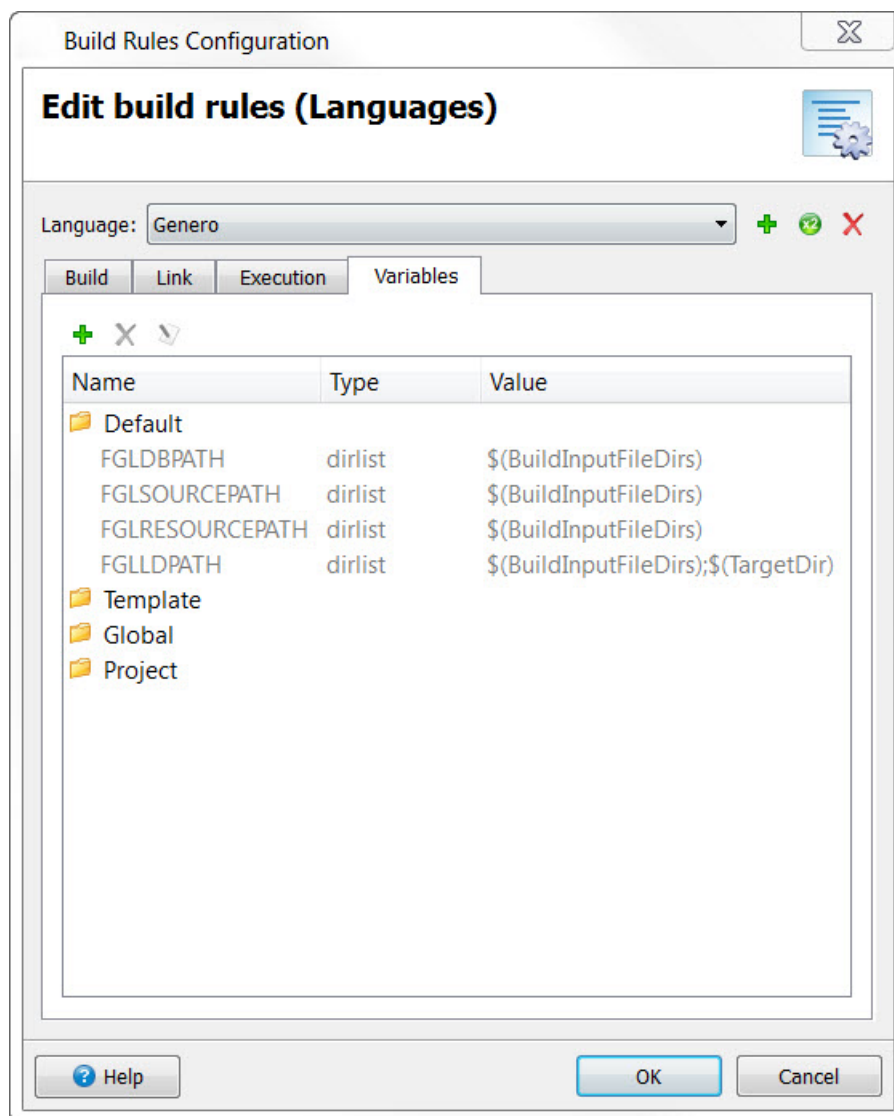


Figure 218: Variables tab

Environment variables

The Variables tab lists environment variables set for the selected language. The integrated Toolbar allows for adding, deleting and editing environment variables.

Add a variable

Select the **Global** category (**Preferences only**) or **Project** category (**Project Manager only**) and click this button to add a new environment variable. See [Add or edit environment variables](#) on page 144.

Delete a variable

Select an existing variable and click the button to remove it.

Edit a variable

Select a variable and click this button to modify variable settings. See [Add or edit environment variables](#) on page 144.

Variable values

See [Predefined node variables](#) on page 366.

Project Manager node properties

Properties can be set for the nodes of a project, to help define the component or specify its behavior.

The Properties view displays the properties for the selected group, application, library node, or file. The assigned value for the property displays. You can add or update the property value, or use the undo button to reset the property to its default value.

Note: See [Package and Directory nodes and properties](#) on page 999 for Genero Mobile packaging and directory node properties.

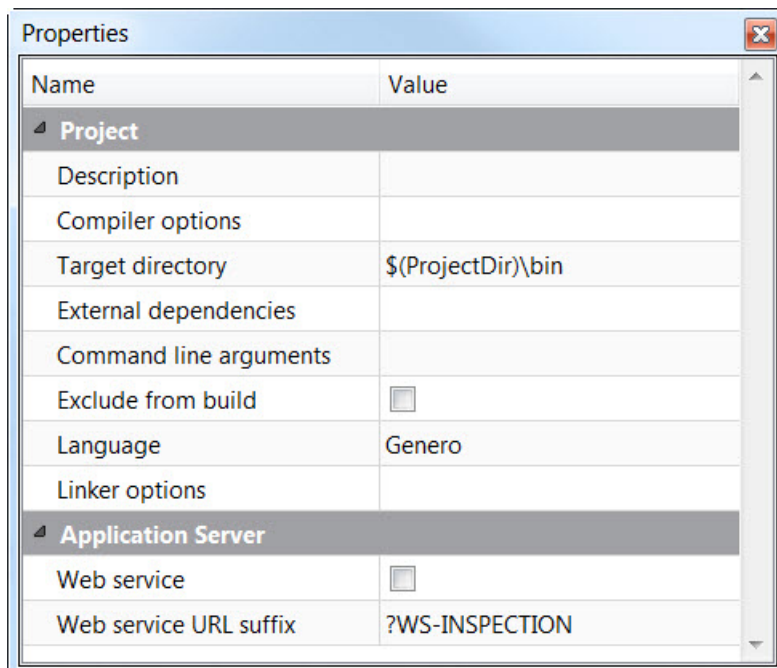


Figure 219: Properties view for an application node

Specific [Project Manager node variables](#) can be used in the values.

Table 104: Project Manager properties

Label	Description	May be inherited	Read-only	Group	Application	Library	File
Description	A short description of the group, application or library node.	NO	NO	YES	YES	YES	NO
Compiler options	Options to be passed to the compiler. For example, -S to dump Static SQL messages found in the source.	YES	NO	YES	YES	YES	YES
Target Directory	Target directory for output files for all applicable nodes. Compiled modules and link results (42m, 42f, 42r, and 42x files) will be stored in this directory.	YES	NO	YES	YES	YES	NO

Label	Description	May be inherited	Read-only	Group	Application	Library	File
	<p>The default directory is a <code>bin</code> directory created in the current Project directory. Use the Browse button to change the directory.</p> <p>If the directory value is changed, Genero Studio tries to maintain a relative path having the project directory as its base. A relative path is not possible, however, if a different drive under Windows™ is selected.</p> <p>Important: For portability, we recommend that Target Directory should always be defined through a relative path, with the Project Directory <code>\$(ProjectDir)</code> as the starting point.</p>						
Source Directory	Directory of source files.	YES	NO	NO	NO	YES	NO
External dependencies	See Setting external dependencies on page 343.	NO	NO	NO	YES	YES	NO
Command line arguments	<p>Arguments passed to the FGL application when it is launched (Run or Debug). This is useful when the application is written to behave differently depending on arguments that are passed on the command line.</p> <p>For example, the value 123 could be entered as the argument. This value could be retrieved in the application source code using the built-in function <code>ARG_VAL</code>, and the application could be written to respond accordingly.</p>	NO	NO	NO	YES	NO	NO
Exclude from build	Excludes the node from the build process.	NO	NO	YES	YES	YES	NO
Exclude from compilation	Excludes the file from compile process.	NO	NO	NO	NO	NO	YES
Exclude from link	Excludes the file from linking process.	NO	NO	NO	NO	NO	YES
Language	Language to be used when building a node. See Languages on page 344	YES	NO	YES	YES	YES	NO
Linker options	Options to be passed when linking.	YES	NO	YES	YES	YES	NO

Label	Description	May be inherited	Read-only	Group	Application	Library	File
File Path	The complete path of a file, including the file name.	NO	YES	NO	NO	NO	YES
Web Service	When checked, indicates the application is a web service application. This property is found under the Application Server group.	NO	NO	NO	YES	NO	NO
Web Service URL suffix	The suffix used to generate the URL when starting a Web Service application. In a Genero Configuration using the GAS as a front-end, the URL generated for a Web Service application is changed to: <i>http://host:port/connector/ws/r/application_name/web_service_suffix</i> where: <ul style="list-style-type: none"> "ws" is used in the URL, instead of the traditional "ua" or "wa". <i>application_name</i> is the name of the application node. <i>web_service_suffix</i> is the value of the associated property. It can be empty. Other values depends on the current GAS configuration. This property is found under the Application Server group.	NO	NO	NO	YES	NO	NO

Predefined node variables

Each Project Manager node has a defined set of variables containing values determined at runtime.

These variables can be used in [Build rules](#) or [Environment variables](#). The variables available on a node are the concatenation of the variables defined for the node plus the ones defined for its ancestors.

Table 105: Predefined node variables (X indicates available for the node)

Variable	Description	Group	Application	Library	File	Package
\$(BAFilePath)	Absolute path of the 4ba file.	X	X	X	X	
\$(BinaryName)	Name of the binary node.		X	X	X	
\$(BuildInputFileDirs)	File directories. List of all files which are input files for a build rule. For example, main.4gl form.4fd test.per toto.4gl.		X	X		

Variable	Description	Group	Application	Library	File	Package
\$(BuildInputFileNames)	File names.		X	X		
\$(BuildInputFilePaths)	Absolute file paths.		X	X		
\$(BuildOutputBaseNames)	Base names. List of all files which are output files for a build rule. For example, main.42m form.42f test.42f toto.42m		X	X		
\$(BuildOutputFileNames)	File names.		X	X		
\$(BuildOutputFilePaths)	Absolute file paths.		X	X		
\$(CommandLineArgs)	Command line arguments.		X			
\$(CompilerOptions)	Compilation flags defined in the user interface.	X	X	X	X	
\$(Dependencies)	List of link output files (link results) of all dependencies binary nodes. For example, your Application depends on 2 libraries, lib1 and lib2. The lib1 link rule creates the \$(TargetDir)/lib1.so output file, and the lib2 link rule creates the \$(TargetDir)/lib2.so output file. The Application Dependencies is a list of these paths: {\$(TargetDir)//lib1.so \$(TargetDir)//lib2.so}		X	X		
\$(DistDir)	Value of the Distribution directory property.					X
\$(ExecutableName)	Executable name.		X			
\$(ExternalDependencies)	Value of the node property External dependencies . Note: For Java™, C, and other languages, this property will contain the full path for external dependencies as FGLLDPATH is not used.		X	X		
\$(GSTDIR)	Genero Studio installation directory path.	X	X	X	X	X
\$(GSTSETUPDIR)	Application Generator template directory path.	X	X	X	X	
\$(Implicit)	FGL argument -implicit=none				X	
\$(InputBaseName)	Input file base name, without extension.				X	

Variable	Description	Group	Application	Library	File	Package
\$(InputDir)	Absolute directory of the input file.				X	
\$(InputExtension)	Input file extension.				X	
\$(InputMimeType)	Input file MIME type.				X	
\$(InputName)	Input file name, with extension.				X	
\$(InputPath)	Absolute path of the input file.				X	
\$(IntermediateFilePaths)	List of intermediate file paths.		X	X		
\$(JavaSourceDir)	Absolute path of the Java source directory (packages root).			X		
\$(JavaSourcePaths)	Java source path (JavaSourceDir of dependencies).			X		
\$(Language)	Value of the Language property.	X	X	X	X	
\$(LinkerOptions)	Value of the node property LinkerOptions		X	X	X	
\$(LinkOutputFileBaseNames)	File names. List of all files which are output files for a link rule. For example, Application.42r		X	X		
\$(LinkOutputFileNames)	File names. List of all files which are output files for a link rule. For example, Application.42r		X	X		
\$(LinkOutputFilePaths)	Absolute file paths		X	X		
\$(PackageName)	Value of the PackageName property.					X
\$(ProjectDir)	The directory where the 4pw file is located, or the operating system temporary directory if the project has never been saved.	X	X	X	X	
\$(RelativeDir)	A list of directory names, corresponding to the path difference between the source directory and the current directory. For example, the source directory is \$(ProjectDir)/src, and the file path is \$(ProjectDir)/src/com/d1/Account.java. The relative directory would be the string list {com d1}.		X	X		

Variable	Description	Group	Application	Library	File	Package
	To use, you typically join with a separator: <ul style="list-style-type: none"> For paths that use the backslash: <code>\$(RelativeDir /)</code> For a java package using a period: <code>\$(RelativeDir .)</code> 					
\$(TargetDir)	Target directory of the compiled files.	X	X	X	X	
\$(TargetPaths)	List of target directories of all dependencies.	X	X	X	X	
\$(XCFFilePath)	Path of the XCF file found for the application.		X	X		

List Expansion

For all variables that contain a list of files, for example `$(BuildInputFiles)`, special syntax is used to expand the list.

To get a list with a specific file extension, use a colon:

```
$(variableName:.extension)
```

For example, `$(BuildInputFiles:.4gl)` returns all files with a 4gl extension in the `$(BuildInputFiles)` list, separated by a space.

To get a list with a specific separator, use the pipe symbol:

```
$(variableName|separator)
```

For example, `$(BuildInputFiles|#)` returns all the files in the `$(BuildInputFiles)` list, separated by a #.

Project Manager error messages

A list of Project Manager error messages. For messages that are not self-explanatory, additional information is provided.

Table 106: Project Manager error messages

Number	Description
GS-12001	<pre>`%1' could not be found on the disk.</pre> <p>The file to open is not present on disk. Check if the file path is correct.</p>
GS-12002	<pre>`%1' is not on the same drive than the project '%2'. The project file will not be portable.</pre>

Number	Description
	The file is not on the same drive as the project, resulting in no relative path between the file and the project. This causes some limitations, for example using versioning will not be possible.
GS-12003	Cannot load '%1' language file Language file loading error. Transform to a generic loading error message with the file path.
GS-12004	Cannot load '%1' platform file Platform file loading error. Transform to a generic loading error message with the file path.
GS-12005	Node '%1' contains orphan properties, clean document settings to remove them User settings were used in the project file, but are not in the current template. Choose the right template (GSTSETUPDIR) or clear the settings (losing the values).
GS-12006	Orphan property %1, clean document settings to remove it User settings were used in the project file, but are not in the current template. Choose the right template (GSTSETUPDIR) or clear the settings (losing the values).
GS-12007	Orphan property group '%1', clean document settings to remove it User settings were used in the project file, but are not in the current template. Choose the right template (GSTSETUPDIR) or clear the settings (losing the values).
GS-12008	GSTSETUPDIR='%1' doesn't exist GSTSETUPDIR is relative or doesn't exist. Check the path in the Genero configuration.
GS-12009	Environment variable '%1' should not be defined in the project but in the config only One invalid environment variable has been set in the project. The following list contains the forbidden variables: <ul style="list-style-type: none"> • GREDIR • GSTSETUPDIR • GSTWCDIR • FGLDIR Move its definition to the Genero Configuration.
GS-12010	Some files are not on the same drive than the project file. The project has been saved with absolute paths and will not be portable. Error message and resolution should be self-explanatory.
GS-12011	Unable to create a node of type '%1'. Internal error: incorrect project file format. A node could not be created.

Number	Description
GS-12012	<p>For external project file format version must be equal or greater than '%1'.</p> <p>Internal error: an external project of an unsupported format has been added.</p>
GS-12013	<p>Unable to write the file '%1'. Check path existence and permissions</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12014	<p>Unknown encoding '%1'.</p> <p>No suitable codec can be found for the specified encoding.</p> <p>Add a new alias in the encoding map or add a new codec charmap.</p>
GS-12015	<p>Cannot load external project '%1', project already loaded.</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12251	<p>The file '%1' is already present in the project and has not been imported</p> <p>The import action tries to add a file which is already present. The file is ignored (warning).</p>
GS-12252	<p>Cannot compute dependencies for function '%1' because it is defined multiple times</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12253	<p>Circular dependency detected on '%1' node</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12254	<p>File is already present in the project</p> <p>This warning signals that one of the saved files is already present in the project and won't be added a second time.</p>
GS-12255	<p>A business application diagram file is already present in the project</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12401	<p>Added dependency on '%1' node to '%2' library</p> <p>Information message: the import process created a dependency.</p>
GS-12501	<p>Unable to find the wsdl compiler. Check your web services installation.</p> <p>Fglwsdl tool is not found in the fgl install.</p> <p>Check if the Genero configuration uses an fglgws VM.</p>
GS-12502	<p>Cannot load external dependency '%1'</p> <p>An error occurred loading an external project (4pw).</p>
GS-12503	<p>Cannot load project database for external dependency '%1'</p> <p>An error occurred loading an external project database (4pwndb).</p>

Number	Description
GS-12505	<p>Project database: <code><ERRORMESSAGE></code></p> <p>The content of error message GS 12505 can vary because it is an error that occurs during the build process of the project manager database, and is dependent on the external tool that is running. This will likely be a compiler (fglcomp) error.</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12509	<p>Cannot create package XML file '%1'</p> <p>An error occurred during <code>\$(GenerateXMLPackage)</code> command.</p>
GS-12510	<p>Unknown argument '%1' to <code>\$(generateXMLPackage)</code> task.</p> <p>The <code>\$(GenerateXMLPackage)</code> command arguments are incorrect.</p>
GS-12511	<p>Unknown property '%1'</p> <p>Some unknown property is defined. This message mostly appears for AG settings.</p>
GS-12512	<p>Load failed</p> <p>General error: an error occurred during file load.</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12513	<p>Unsupported version</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12514	<p>Missing version</p> <p>An error has occurred with the <code>pm-settings.conf</code> file in the template directory <code>\$(GSTSETUPDIR)</code> because the version attribute is missing. Note : this message may occur with other files using a version attribute.</p>
GS-12515	<p>Unknown node '%1'</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12516	<p>Platform '%1' is not defined</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12517	<p>Language '%1' is not defined</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12751	<p>Configuration isn't valid. Operation canceled.</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12753	<p>Cannot create execution task.</p> <p>This is caused by an internal system error.</p> <p>Contact your local Four Js support center.</p>
GS-12754	<p>Cannot create profiling task.</p> <p>This is caused by an internal system error.</p> <p>Contact your local Four Js support center.</p>
GS-12755	<p>Cannot create debugging task.</p>

Number	Description
	<p>This is caused by an internal system error.</p> <p>Contact your local Four Js support center.</p>
GS-12756	<p>The debugger is already running. You have to stop the current debugger session to start a new one.</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12757	<p>Operation cannot be performed, no item selected</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12758	<p>Unable to find the internal FGL installation. Check your %1 installation.</p> <p>Internal error: Genero Studio install malfunctioned.</p> <p>Reinstall Genero Studio to solve the problem.</p>
GS-12759	<p>Unable to start an import on the selected node.</p> <p>Internal error: the import does not work on the currently selected node (only the main project and group are supported).</p>
GS-12760	<p>Unable to build an unsaved project using a remote server. Unsaved projects can only be built with local FGL installations</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12761	<p>Genero installation isn't valid. Operation canceled.</p> <p>Error message and resolution should be self-explanatory.</p>
GS-12762	<p>Front end isn't valid. Operation canceled.</p> <p>Error message and resolution should be self-explanatory.</p>

Code Editor

Code Editor is a programming-oriented editor. In addition to editing source code, it can handle any kind of text or languages such as 4GL and XML. Smart editing features like auto-completion, code templates, text folding, bookmarking, and robust search and replace make coding easier and more efficient.

- [Editing code files](#) on page 374
- [Using the Diff tool](#) on page 380
- [Printing files](#) on page 382
- [Using XML catalog files](#) on page 383
- [Code Editor Reference](#) on page 384

Editing code files

Information about editing files in Code Editor.

- [Code Editor basics](#) on page 374
- [Smart editing - indenting, tabs, and backspace](#) on page 375
- [Fold text](#) on page 375
- [Bookmarks](#) on page 376
- [Auto completion \(Ctrl+Space\)](#) on page 376
- [Code templates \(Ctrl+T\)](#) on page 376
- [Split a document](#) on page 377
- [Square selection](#) on page 377
- [XML editing](#) on page 377
- [Search and replace](#) on page 377

Code Editor basics

The active document is displayed in the Document workspace. Multiple documents can be open, in tabbed windows, with the filename on the tab.

By default, **Line numbers** are displayed at the left side of the window.

The **status bar** at the bottom of the window contains the **cursor position** (line, column), the format (Windows™, UNIX™, MAC), and the **mode** (insert/overstrike).

The [Code Structure view](#) displays information about the structure of the active file. Clicking on an element in the Structure view will display and highlight the corresponding lines in the Editing window.

vi Editor

You can change the editor to use vi commands. Select **Edit >> VI Editing Mode** to switch the editor.

Syntax highlighting

The elements of the program are visually highlighted. Language key words, strings, variables and comments are each colored differently, making the program structure easier to understand. Select **Tools >> Preferences, Code Editor** to customize the behavior and color.

Syntax errors

An error mark in the gutter flags syntax errors as they occur. Select the error mark to display a message concerning the error. The error message and line number also display in the **Document Errors** tab in the output.

Menu of options

Use the Edit menu or right-click in document to display some options for selecting, searching, and editing. The available accelerator keys are documented in the [Keyboard Shortcuts](#) on page 394 page.

Document Format

You can change the document format (Windows™, UNIX™, MAC) by selecting **Edit >> Convert to**.

Integrated diff

The [Using the Diff tool](#) on page 380 tool is integrated into Code Editor.

Smart editing - indenting, tabs, and backspace

Code Editor has smart editing features for indenting, tabs, and backspace.

Smart Indent

The Smart Indent feature is enabled by default. After you press the Enter key, the new line is indented to align with the immediately preceding non-blank line.

For example:

```
01 LET var1 = 55
02 LET var2 = 22
```

If you do not wish your new line to be aligned with the preceding line, use the Left Arrow key to move the cursor towards the left margin of the page.

Smart Tabs

The Smart Tabs feature is enabled by default. When the Tab key is pressed at the beginning of a line, the cursor is aligned with the first character following the next whitespace on the immediately preceding non-blank line.

For example:

```
01 DEFINE
02   var1 INTEGER,
03   var2 INTEGER
```

If the preceding line does not have embedded whitespace, the Tab key moves the cursor the number of spaces indicated by the **Tab size** setting in the [Behavior and display preferences](#).

Smart Backspace

The Smart Backspace feature is enabled by default. Pressing the Backspace key will move the cursor back in the current line to the position of the indent in the previous line.

Disable Smart features

Disable Smart features by selecting **Tools >> Preferences >> Code Editor, Behavior & Display** from the menu.

Fold text

Folding condenses portions of text based on the scope of statements.

Click the + or - symbols in the left gutter to fold or unfold all corresponding lines. Use the **View** menu or right-click context menu options to **Fold**, **Fold all**, or **Unfold all**.

Bookmarks

Bookmarks are used to mark areas of a document for easy access.

To add a bookmark, place the cursor in the line of text where you want to add the bookmark. Right-click and select **Toggle bookmark** from the menu. A bookmark icon will appear in the gutter. A list of bookmarks is kept in the Bookmarks view.

Auto completion (Ctrl+Space)

The auto complete feature helps complete a line of code or prompts for a valid keyword in the syntax.

Type the first letters of a word and then press Ctrl-space to complete the word or select a word from a list of options.

Code templates (Ctrl+T)

Code templates are snippets of frequently used code elements available to insert into the code to avoid repetitive typing and speed up coding.

To use a code template in your code, select one of these methods:

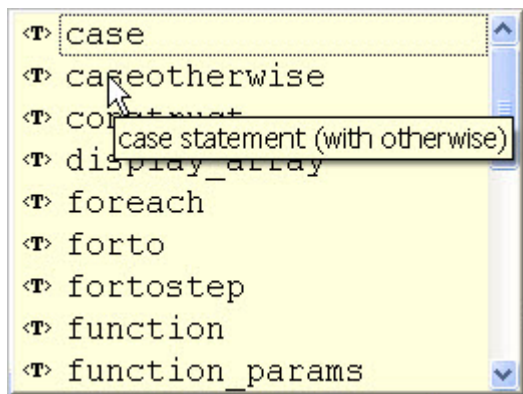
- Type the code template name, such as "**case**", in your code where you want it to appear; Press **Ctrl+T**, or right-click to display the contextual menu and choose **Expand Template**.
- Position the cursor where you want the code to appear, and press **Ctrl+T** to display a list of code templates. Select from the list using the arrow keys, and press Enter. (Press Esc to cancel)

The selected template is inserted in the document using the exact layout (tabs, linefeeds, and so on.) A corresponding number of lines are shifted.

Templates are user-customizable. Use **Tools >> Preferences >> Code Editor, Code Templates** to add or modify a template.

Example: Adding a "case" instruction code template

Use the popup window (Ctrl-T) to select a keyword, or type "case" and press Ctrl-T.



The code template is inserted in your code, ready for you to complete.

Figure 220: Code template pop-up window


```

87:
88: CASE ()
89:   WHEN
90:   WHEN
91: END CASE
92:
93:
94: RETURN fetch_ok

```

Figure 221: CASE code template

Split a document

Splitting the document view allows you to see different parts of a long document at the same time.

Split/unsplit a document using the **View** menu option. Once a split is requested, the current document window splits into two separate windows which can be scrolled independently. Each window may be split again twice.

Highlight rules are applied to the same text in every window pane. For example, placing the cursor on the IF statement in one window also highlights the same IF statement in other windows in which it is displayed.

Square selection

Press the alt key while selecting text with the mouse to select a rectangle of text, instead of entire lines.

XML editing

Code Editor recognizes XML documents and provides XML validation. Color coding, Smart Editing, and Code Completion features can be used with an XML file.

Search and replace

Information about using search and replace in Code Editor.

- [Using wildcards in search](#) on page 377
- [Using regular expressions in search](#) on page 378
- [Group capture in regular expressions](#) on page 379

Using wildcards in search

The **Use Wildcards** option will find the specified combination of characters, including combinations within a word.

Table 107: Use Wildcards

Wildcard	Description	Example expressions	Matches any combination of characters
*	Substitutes for any number of characters	<ol style="list-style-type: none"> 1. st*e 2. *rec 	<ol style="list-style-type: none"> 1. that begins with st, followed by zero or more characters, followed by e: custrec, store, steady 2. that begins with zero or more characters, followed by rec: gr_custrec, RECORD
?	Substitutes for a single character	<ul style="list-style-type: none"> • st?re 	<ul style="list-style-type: none"> • that begins with st, followed by a single character, followed by re:store

Using regular expressions in search

The Search tool will search for an exact match to text in the Find box, unless you specify match conditions using regular expressions, special meta characters, and predefined regular expressions.

Entering the string *FUNCTION*, will find *FUNCTION* or *function*, but not the string *fun*. If the Search option **case sensitive** is checked, the search will distinguish between uppercase and lowercase letters.

Meta Characters

- The meta character `\` matches the character following it, except when followed by a left or right round bracket, a digit 1 to 9, or a left or right angle bracket.
- The special characters "]" and "-" have no special meaning if they appear as the first characters in the set.

Table 108: Meta Characters

This table provides the wildcard symbol or syntax, a description, an example expression, and examples of strings that match the example expression. For some entries, more than one example expression - string match pair are provided.

Wildcard	Description	Example expressions	Matches any string
.	Substitutes for any single character	• fla.	• containing four letters that begin with <i>fla</i> : <i>flag</i> , <i>FLAG</i> , <i>flannel</i>
		• b.g	• containing three letters in the format <i>bxg</i> : <i>big</i> , <i>bog</i> , Bog , <i>bag</i>
*	Substitutes for zero or more occurrences of the preceding expression/character	• a *b (notice the blank before the *)	• "a" followed by zero or more blanks then "b" "a basic" "abasic"
+	Substitutes for one or more occurrences of the preceding expression/character	• a +b (notice the blank before the +)	• "a" followed by one or more blanks then "b" "a basic" "a basic"
\	Searches for the character following; this cancels the special significance of the meta characters including itself, allowing a search for them. When used in a set, it is treated as an ordinary character.	• \+100	• containing +100; treats + as an ordinary character
		• \\user	• containing \user; treats \ as an ordinary character
[set]	Defines a set of characters enclosed in square brackets ([...]) to be used for matching; may define character ranges, as in [a-z] and [0-9]. If the first character in the set is "^", it matches	• [bd]og	• containing <i>bog</i> , <i>dog</i>
		• [Tt]ooltip	• containing <i>Tooltip</i> , <i>tooltip</i>
		• b[^o]g	• containing three characters, <i>b</i> , <any

Wildcard	Description	Example expressions	Matches any string
	any character NOT in the set.		character but <i>o</i> , <i>g</i> : <i>bag</i> , <i>big</i>
		<ul style="list-style-type: none"> [A-Da-d]+ 	<ul style="list-style-type: none"> containing one of the alpha characters <i>a</i> through <i>d</i> inclusive, in uppercase or lowercase: <i>define</i>, <i>Define</i>, <i>age</i>
x y	Matches either expression <i>x</i> or expression <i>y</i> (composite expression)	<ul style="list-style-type: none"> bob bog 	<ul style="list-style-type: none"> containing either string: <i>bob</i> or <i>bog</i>
xy	Strings multiple expressions together, finding a single string containing expression <i>x</i> and expression <i>y</i> (composite expression)	<ul style="list-style-type: none"> def.* iti.* 	<ul style="list-style-type: none"> containing the string <i>def</i>, any characters including none, and then the string <i>iti:definition</i>
^ \$	Restricts the pattern matching to strings at the beginning of the line (^ character) and/or the end of the line (\$ character),	<ol style="list-style-type: none"> ^when test.\$ 	<ol style="list-style-type: none"> with <i>when</i> at the beginning of the line with <i>test.</i> at the end of the line.

Group capture in regular expressions

When the **Use Regular Expressions** option is checked, group capture allows you to isolate groups in the expression to be matched, so they can be captured and substituted during the replacement.

Given this expression in your document:

```
4+5*6+88
22+4*555
```

You can transform this express to these patterns using group capturing: (4+5)*(6+88) and (22+4)*555.

In the first expression there are four groups. First you must capture the individual groups in the expression by enclosing them in parenthesis. Specify that the characters are integers using the regular expression [0-9]+ (one or more integers). Use the escape character \ to indicate that the * and + symbols are literal and not meta characters.

Find text: ([0-9]+\)\+([0-9]+\)*\([0-9]+\)\+([0-9]+)

Replace expression indicating the desired pattern and the positions of the groups (numbered from left to right): (\1+\2)*(\3+\4)

Result: (4+5)*(6+88)

Function search

Function search allows you to search a file for a function.

Display the Function Search view with **Window >> Views >> Function Search**.

Enter a part of the function name to return all functions in all files in the project that meet the search criteria. For example, entering *ord* returns a list of functions with *ord* in their name such as *order_new()* and *close_order()*.

Double-click on a line in the results to go to the selected function.

Using the Diff tool

The *Diff* tool compares two files: a read-only base copy of the file and a working copy. It is integrated into Code Editor.

Comparing files

The Diff tool automatically selects the Diff base file and flags differences between the base file and the working copy with color-coded markups in the Code Editor gutter. Colors can be changed in [preferences](#).

- Green - added lines
- Orange - modified lines
- Red - deleted lines

The base file is selected in this order:

Use Generated File

The default for generated files, this option compares the document with the file generated by the application generator. This is the generated file before POINTs and BLOCKs are injected into the file. This option is made available only if the file has been generated.

Use Repository File

The default for versioned files, this option compares the document with the file in the repository for the current SVN version of the file. This option is enabled only if the file is versioned.

Use File on Disk

Compares the document with the contents of the file as it was on disk when the file was opened. If you select this option again after the file has been opened, the diff data will be refreshed by comparing the document with the actual content of the file on disk.

You can change the base file used with **Diff >> Base File**.

Specify how the tool compares white space and case in [preferences](#).

Display modes

You can select a Diff display mode from the **Diff** menu.

Normal View

This view opens the file in normal editing mode.

Diff View

This view displays differences between the base file and working copy in a single pane document. This is also called *single pane diff mode*. New and modified lines are clearly marked by color. This view identifies the location of deleted lines, but does not display them.

Diff View with Deleted Blocks

This view displays differences between the base file and working copy in a single pane document with deleted blocks highlighted in red.

Vertical Dual Diff View

This view displays the two files vertically, with a left and right pane. It opens the base file (read-only) in

the left pane and the editable working copy in the right pane. This is also called *two pane diff mode*.

Horizontal Dual Diff View

This view displays the two files horizontally, with a top and bottom pane. It opens the base file (read-only) in the top pane and the editable working copy in the bottom pane. This is also called *two pane diff mode*.

Example: Diff View

This figure shows **Diff View**. Lines 28-32 are marked with green blocks to indicate newly added lines. The orange block in the gutter of line 24 indicates a modified line (a new comment, highlighted in green, has been added to the existing line). The red line between line 22 and line 23 flag the location of a deleted line/lines.

Figure 222: Diff View display mode

```

15 IF query_ok THEN
16     CALL fetch_rel_acct(1)
17 ELSE
18     MESSAGE "You must query first"
19 END IF
20 ON ACTION Previous
21     IF query_ok THEN
22         CALL fetch_rel_acct(-1)
23     END IF
24     ON ACTION DELETE -- verify with user
25     IF (delete_check()) THEN
26         CALL delete_account()
27     END IF
28     ON ACTION ADD
29     IF (inpupd_account("A")) THEN
30         CALL add_account()
31         LET query_ok = FALSE
32     END IF
33     ON ACTION Modify

```

Example: Vertical Dual Diff View

This figure shows **Vertical Dual Diff View**. Side-by-side comparison of the base file in the left pane and the editable working copy in the right pane give you a before and after record of changes. Red blocks in the gutter of the base file pane show the contents of lines deleted in the working copy.

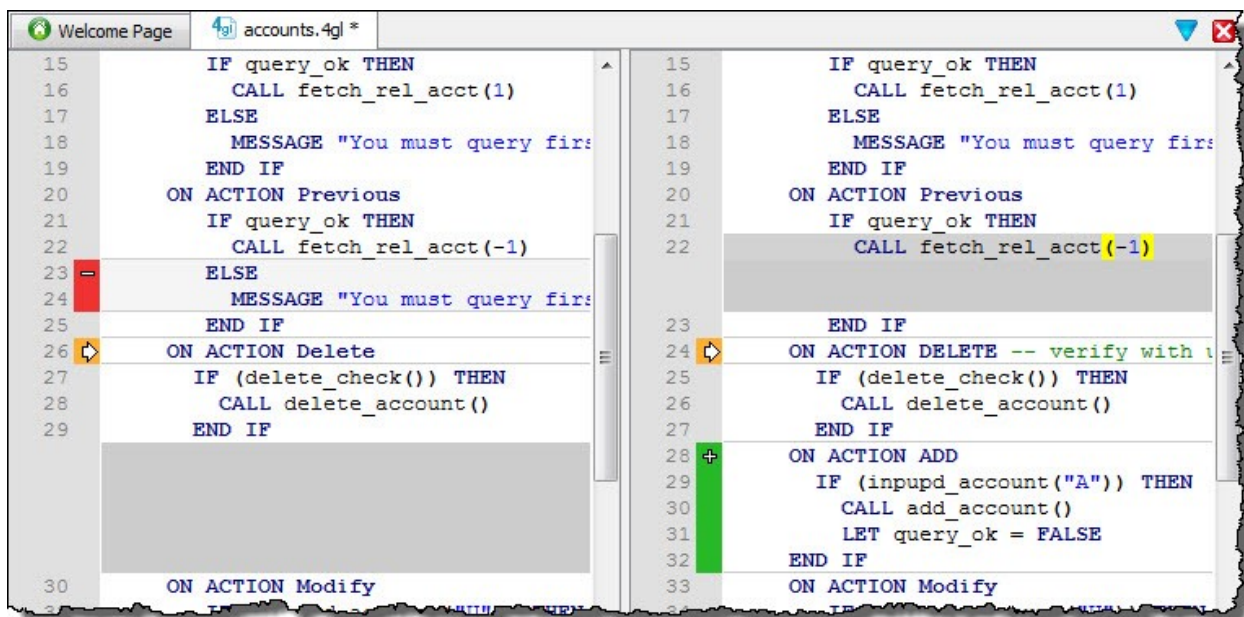


Figure 223: Vertical Dual Diff View display mode

Diff navigation

Locate the changes quickly with Diff navigation options.

First Difference	Locate the first difference.
Previous Difference	Locate the previous difference.
Next Difference	Locate the next difference.
Last Difference	Locate the last difference.

Reverting a difference

To revert a difference, place the cursor in the line of marked code that you want to revert. Select **Diff >> Copy to Right**. The **Copy to Right** action replaces the difference in the working copy with the original values from the base copy. Select **Diff >>Copy All to Right** to revert all changes.

Use an external tool

You can specify an external tool instead of the integrated one to compare files in [preferences](#). The **Tools >> Diff** option will launch the external tool.

Command line option

Use `generostudio -diff file1 file2` at the command line to open the given files in Diff mode.

Printing files

There are two options for printing files in Code Editor.

File >> Print	Prints the file using the options available to your operating system printers.
File >> Print Preview	Displays a preview of the printed file in a preview window. For diagrams, scaling and number of pages to use to print can be configured.

Using XML catalog files

Information about XML catalog file usage in Code Editor.

- [XML catalog files](#) on page 383
- [The XML catalog file](#) on page 383
- [Manage XML catalog entries](#) on page 384

XML catalog files

XML Catalog files are used to provide an alternative path when the external entities are not accessible, or are not where the XML document specifies.

Many XML documents contain external links to stylesheets, schemas, DTDs, and so on, which may be stored on remote systems. If the links are absolute URLs, they only work when your network can reach them. However, the entity resolver of the XML SAX parser can be used to determine whether there is a local equivalent in your system's **Catalog** application level cache.

When you use an XML Catalog file to map these external references to local equivalents, the task of locating the reference is shifted from the XML documents to the XML Catalog files, which can be varied for different audiences. Since local copies of the references are accessed instead of remote network resources, and the local copy can be a subset of the complete schema or DTD in the external reference, XML processing may be faster.

The [XML catalog files](#) to be used by Genero Studio are specified through the [General Preferences](#) on page 106.

The XML catalog file

Genero Studio provides a subset of the OASIS XML catalog specification.

- The **catalog** element to assign a name to the catalog
- **System** and **public** elements to resolve DTDs
- The **uri** element to resolve schema location for XSDs
- The **nextCatalog** element to refer to another catalog file

Example File

This example illustrates the features of a Genero Studio XML Catalog file:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--
  FOURJS_START_COPYRIGHT(D,2006)
  Property of Four Js*
  (c) Copyright Four Js 2006, 2011. All Rights Reserved.
  * Trademark of Four Js Development Tools Europe Ltd
  in the United States and elsewhere

  This file can be modified by licensees according to the
  product manual.
  FOURJS_END_COPYRIGHT
-->
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
  1<public publicId="-//W3C//DTD XHTML 1.1//EN"
    uri="D:/xml catalog/xml files/xhtml11-flat.dtd"/>
  2<system systemId="docbook4.5.dtd"
    uri="D:/xml catalog/docbook/docbook-V4.5/docbook4.5.dtd"/>
  3<system systemId="http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd"
    uri="D:/xml catalog/docbook/docbook-V4.5/docbook4.5.dtd"/>
  4<uri name="http://www.4js.com/ns/gas/2.32/cfas.xsd"
    uri="D:/xml catalog/xml files/cfas_1.xsd"/>
  5<nextCatalog catalog="C:/Program Files/FourJs/Genero Studio 102327/gst/confcatalog-gst-defaults.xml"/>
  6<nextCatalog catalog="C:/Program Files/FourJs/Genero Studio 102327/gas/etc/catalog"/>
</catalog>

```

Figure 224: Genero Studio XML Catalog file

The **catalog** element is the root of an XML Catalog. The **xmlns** attribute assigns a name to the catalog file.

Additional Elements in this file are:

1. **public** - resolves a Public ID for an external DTD; specifies Public external DTDs intended for broad use (public distribution of a DTD file for a wider audience).
2. **system** - resolves a relative system id for an external DTD; used to find the public DTD if it cannot be located by the public ID.
3. **system** - resolves a web-based system id for an external DTD.
4. **uri** - used to locate XSL and other files. It can be used for everything that is not a declared PUBLIC or SYSTEM identifier for a DTD or system entity file.
5. **nextCatalog** - resolves entries related to [Additional Catalogs](#).

Manage XML catalog entries

You can add, edit, and delete XML catalog entries.

Use **Tools>>Preferences** and select **XML Schema/DTD** from the **Code Editor** page in the Pages tree; a list of the XML catalog files to be used with Genero Studio is displayed,

You can add/edit/delete/edit entries for these catalog files through [XML Schema/DTD Configuration](#).

Code Editor Reference

Reference information for Code Editor.

- [Code Editor preferences](#) on page 384
- [Views](#) on page 390
- [Keyboard Shortcuts](#) on page 394
- [vi Commands List](#) on page 398

Code Editor preferences

Set preferences for the Code Editor.

Select **Tools >> Preferences, Code Editor** to access Code Editor preferences.

Shortcut keys can be customized with **Tools >> Preferences, User Interface, Accelerators**.

VI Mode Settings

Table 109: VI Mode Settings options

Option	Description
Open file with VI mode editing check box	If checked, the behavior of the editor is modified to behave like a UNIX™ editor.

Diff Mode Settings

Table 110: Diff Mode Settings options

Option	Description
Default diff mode combobox	<p>Set the default diff mode. This is the mode in which Code Editor will open files. This mode has no effect when comparing two files on disk; dual pane will always be opened in this case.</p> <p>No diff Open in normal editing mode.</p> <p>Single pane Open in diff mode in a single pane view.</p> <p>Single pane with deleted blocks Open in single pane with deleted blocks highlighted.</p> <p>Dual pane Open in two panes with the base file on the left and the file to compare on the right.</p>

- [Encoding \(i18n\)](#) on page 385
- [Behavior & Display preferences](#) on page 385
- [Color and font preferences](#) on page 387
- [Template preferences](#) on page 387
- [XML Schema/DTD preferences](#) on page 388

Encoding (i18n)

Text Editor supports internationalization (**i18n**), allowing you to enter special characters like Japanese, Chinese, or Indian scripts, and to enter text from right to left as in Arabic.

The characters typed at the keyboard are intercepted and changed automatically, based on the Encoding method selected in [General Preferences](#) on page 106. The default Text Encoding is **Default charset**.

Behavior & Display preferences

Set preferences for the behavior and display aspects of the Code Editor.

Language

Select the programming language to which the preferences will apply; the default is **All Languages**. If you select a specific language, the changes will apply only to that language.

Use custom settings - this checkbox appears only if Language is not set to **All Languages**. This box must be checked to make any changes in the settings for the selected language.

In order to enable the [Language-Specific tab](#), Language must be set to a specific language.

Behavior Preferences

Tabs:

- **Tab size** - default size is four characters.
- **Insert spaces for tab-** if checked, *tab size* whitespaces are inserted into the document instead of the tab character when the Tab key is pressed.

Indentation:

- **Smart tab** - if checked, when the Tab key is pressed the cursor is moved to align with the first character following a whitespace on the previous non-blank line.
- **Smart backspace** - if checked, backspace intelligently through whitespace.
- **Smart indent** - if checked, indent code based on the indentation of the previous line. New lines are indented to the first non-blank character of the line above.
- **Strip trailing white spaces** - remove trailing whitespace from the file

Automatically Close: If checked, the closing symbol will be added to each of these:

- **String**
- **Single quotes**
- **Parenthesis**
- **Curly braces**
- **Square brackets**
- **Angle brackets**

Smart Key Options:

- **Smart home** - if checked, the Home key moves the cursor to the left of the first non-blank character on the line.
- **Smart end** - if checked, the End key moves the cursor to the right of the last non-blank character on the line.

New Document Format: Sets the default line ending for new text documents:

- **Windows** - Use the CRLF end-of-line format, as used by Windows and most other early non-Unix and non-IBM operating systems.
- **UNIX/Mac** - Use the LR end-of-line format, as used by UNIX-like systems including Mac OS X.
- **Mac 9** - Use the CR end-of-line format, as used by Mac OS up to version 9.

Display Preferences

Editor:

- **Show line numbers** - Enables or disables line number display
- **Show Right Margin** - Displays a thin line on the right side to indicate the right margin
- **Right margin** - Row number where the right margin line is located. Sets the right margin of the editor. The default is 80 characters
- **Show indentation guide** - vertical lines that indicate relative indentation of text

Highlighting:

- **Current line** - highlight line at caret position
- **Brace Match** - enable brace match highlighting (highlights both braces in a set when the cursor is immediately to the right of one of the braces).

Wrapping:

- **Line wrapping** - enabled if checked.

Language Specific Preferences

When [Language](#) is not set to All languages, the **Language Specific** tab is enabled. Click the tab to display any preferences related to that language.

Color and font preferences

Set preferences for the behavior and display aspects of the Code Editor.

You can specify how specific styles of your Editor code appear.

Global Style - applies to all languages

- **Theme** - quickly configure the display using predefined color combinations
- **Font family** - specify the font family; when this is changed it applies to all languages, except those for which [custom settings](#) have been created.
- **Font size** - specify font size, when this is changed it applies to all languages, except those for which [custom settings](#) have been created.

Style Specific Settings (custom settings) for all Languages

Language - the default selection **All Languages** in the combobox displays the default styles and properties applicable to all languages. As a default, specific language pages inherit the settings defined for **All languages**.

Style - list of styles. Select a **style** to display these properties:

- **Color** - specifies the color for the **Foreground** and **Background** of the style, when this is changed it applies to all languages.
- **Font** - specifies the font selected from the available screen fonts installed on your system. **Use custom font for the selected style** must be checked in order to make changes in:
 - **Family** - font family associated with your selected font; changes will override the global setting.
 - **Size** - font sizes associated with your selected font. changes will override the global setting.
- **Bold** - specifies whether the selected font is Bold. Inherits the global family and size.
- **Italic** - specifies whether the selected font is Italic. Inherits the global family and size.

Style Specific Settings (custom settings) for a specific Language

Language - list of languages. Select the **language** from the dropdown list, to display the language-specific styles and properties.

Use custom settings - this checkbox displays when you select a specific language in the list. This box must be checked in order for you to make any changes in the settings for the selected language. When this box is checked, the settings for the selected language become specific, and it does not inherit the **All languages** settings any longer.

- To change the **Color (Background, Foreground)** or the font properties **Bold** or **Italic** for a specific style, you must first select the **Style** from the Style list. **Use custom settings** must be checked.
- To change the **Font Family** and **Size** properties for a specific style, you must check **Use custom font for the selected style**. This overrides the Global Font family and Font size settings.

Template preferences

Set preferences for the template aspects of the Code Editor.

Language - select the desired language. The default is Genero BDL.

Templates

Displays a list containing the **Name** and **Description** of existing templates. When an existing template is selected, the corresponding template code is displayed in **Preview**.

Icons allow you to:

- **Add** - Displays the **Add Code Template** dialog box, allowing you to enter the new template name and description
- **Delete** - Deletes the selected template
- **Edit** - Displays the **Edit Code Template** dialog box, allowing you to edit the name and description of the selected template.

Add/Edit a Template:

- For new templates, enter the template name and description.
- Type directly in the pane to add the code for a new template or to make changes in existing code.
- Use the pipe character '|', to denote the cursor position in the expanded template. For example, when you insert this template into your code, the cursor appears inside the parenthesis after the CASE keyword.

```
CASE ( | )
    WHEN ( )
    WHEN ( )
END CASE
```

Important: When you modify or create a template, you must type it exactly. For example, you must use indents.

Preview - Displays the code for the selected template.

Import/Export buttons

- **Import** - import 4GL template definitions from an XML file
- **Export** - export 4GL template definitions to an XML file

XML Schema/DTD preferences

Set preferences for the XML schema and DTD aspects of the Code Editor.

Many XML documents contain external links to stylesheets, schemas, DTDs, and so on, which may be stored at remote locations. [Genero Studio XML catalog files](#) may be used by Genero Studio to provide an alternative path when the external entities aren't accessible.

Use **Tools>>Preferences** and select **XML Schema/DTD** from the Code Editor page in the Pages tree. A list of the XML catalog files provided with Genero Studio is displayed.

XML Schema/DTD Configuration

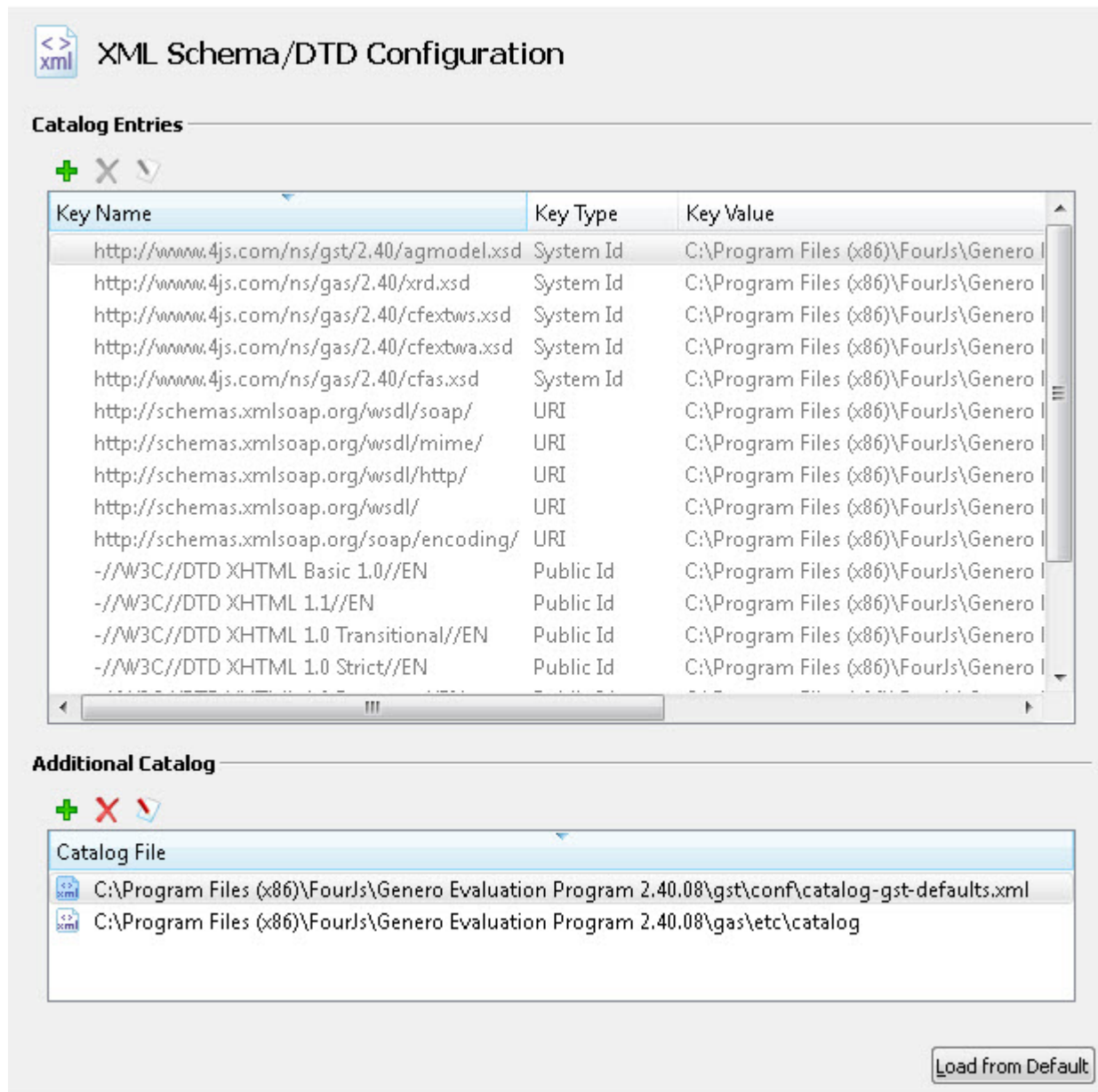


Figure 225: XML Schema / DTD Configuration

Catalog Entries

This section displays the entries for the XML Catalog files that are provided by Genero Studio.

Use the icons at the top of the Catalog Entries section to add/delete or edit catalog entries. An edit icon appears next to catalog entries that may be edited.

These values are displayed for each Catalog Entry:

- **Key name** - can be empty; already existing names can not be used.
- **Key type** - valid values are:
 - **Public ID**, for the DTD
 - **System ID**, for the DTD
 - **Schema URI**, for the Schema location

- **Key value**
 - For **Public/System IDs** - the absolute file location for the DTD
 - For **Schema URI** - the XSD file location; a Browse button is provided to locate the grammar file.

Additional Catalog

Entries in this section specify an XML catalog file to be added to the end of the current catalog. This allows one catalog to refer to another. If a reference cannot be resolved in the current catalog entry file, then Genero Studio moves to the next catalog specified in the Additional Catalogs section.

Entries defined in a [Genero Studio XML Catalog file](#) are given preference over entries that come from a Next Catalog file. This allows you to override the entries in the Next Catalog file to resolve external grammar files.

Use the icons at the top of this section to add/edit/delete Additional Catalog entries.

- **Next Catalog File** - specifies the name of the additional XML catalog file.

See [Using XML catalog files](#) on page 383 for additional information about Catalog files.

Customize Diff tool: preferences

Select **Tools >> Preferences, Diff** to set preferences for the Diff tool.

Internal tool configuration

Select the checkbox to enable an option:

- Ignore case
- Ignore all white space
- Ignore space change
- Try hard to find a smaller set of changes

External tool configuration

If you have chosen to use an external tool, enter the configuration information:

External tool command line

The path of the external tool

External tool arguments

Use the shown variables to specify the arguments for the command line. The names of the files will replace the variables when the tool is invoked.

Color

Select the color button next to each option to change the color used for deleted, modified, added lines and placeholders.

Views

Information about Code Editor views.

- [Code Editor basics](#) on page 374
- [Code structure view](#) on page 391
- [Output view](#)
- [The Search/Replace view](#) on page 391
- [The Search Results view](#) on page 393
- [Bookmarks view](#) on page 97

Note: For SVN-related views, see [Source Code Management - SVN](#) on page 529.

Code structure view

If the file being edited is a Genero source code file (4gl), its structure is displayed in a tree.

- **Module listing** - the program blocks and functions, together with their variables
- **SQL** - cursor names
- **Globals** - the global variables, including records
- **Externals** - functions from another 4gl file that are called in the active file

Click an object in the tree to display the corresponding lines in the Document window.

The functions listed in the structure are automatically displayed in the order in which they appear in the file.

Right-click in the **Code Structure view** to display a menu of options:

- **Sort alphabetically/Sort by file order** - switch the order in which the functions, and the variables within the functions, are listed in the view. The default is "file order", the order in which the functions and variables appear within the file.
- **Sort ascending/descending** - specify order

The Search/Replace view

The Search/Replace view allows you to search for and replace text in one or multiple documents.

Using the **Window >> Views >> Search/Replace** menu option opens the Search/Replace View:

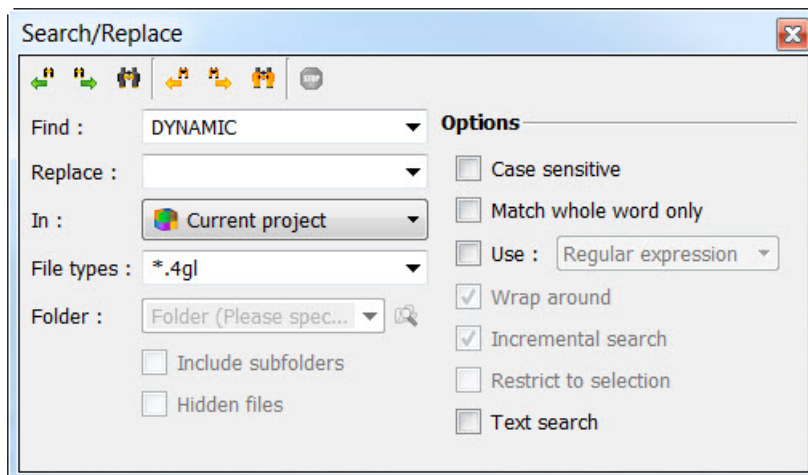


Figure 226: Search/Replace View

Fields allow you to set the search / replace criteria:

Find

Enter a search string, or click the down arrow next to the input box to select from a list of previously entered strings.

Replace

Leave this empty, unless you want to replace the text that is found. You can enter the replacement string, or click the down arrow next to the input box to select from a list of previously entered replacement strings. To replace the text with a space, type a space in this box.

In

Select the scope of the search/replace:

- **Current file** - Search in the current file only.
- **Open files** - Search in all open files.
- **Folder** - Search within a specified folder. You specify the folder, as well as some additional options, in the **Folder** field. Selecting **Folder**

has the same effect as the **Find in Files** menu option.

- **Current project** allows you to search the file content of the project for the search string.

File types

Limit your search and replace to specific file types, specified by * .*ext* where *ext* is a file extension.

Folder

Limit your search results to the contents of a specified folder. You can use the magnifying glass icon to specify the folder. Within this folder, you can use the check boxes to identify whether to include the subfolders of the selected folder, and whether to include hidden files.

Note: The **Folder** combobox is enabled when Folder is selected for the **In** field.

Options allow you to further refine your search / replace:

Case Sensitive

Differentiates between uppercase and lowercase when performing a search.

Match whole word only

Does not find and/or replace the string if it is contained within another string.

Use

Choose between [Regular Expressions](#) or [Wildcards \(* and ?\)](#).

Note: If **Use** is checked, the **Match whole word only** option can not be used and is disabled.

Wrap around

If the search reaches the end of the file, wrap around the document to find the next occurrence.

Incremental search

Stop at each occurrence of the string

Restrict to selection

If a section of a document is selected, restricts the search to that section.

Text search

Search file as text, not as formatted Genero file.

In the view's toolbar, icon buttons allow you to control the direction of the search and the implementation of the replace. The icons in order (left to right):

- Find previous
- Find next
- Find all
- Replace previous
- Replace next
- Replace all
- Stop

Use **Esc** to return focus back to the current open file.

The **Edit >> Search/Replace** provides a menu interface that allows you to perform many of the same tasks allow you to find/replace text in documents, as well as clearing the Search Results view.

- **Edit >> Search/Replace >> Find** (CTRL + F) opens the Search/Replace view.
- **Edit >> Search/Replace >> Find in Files** (CTRL + SHIFT + F) opens the Search/Replace view with Folders selected for the **In** field.

The Search Results view

Search results are displayed in the Search Results view.

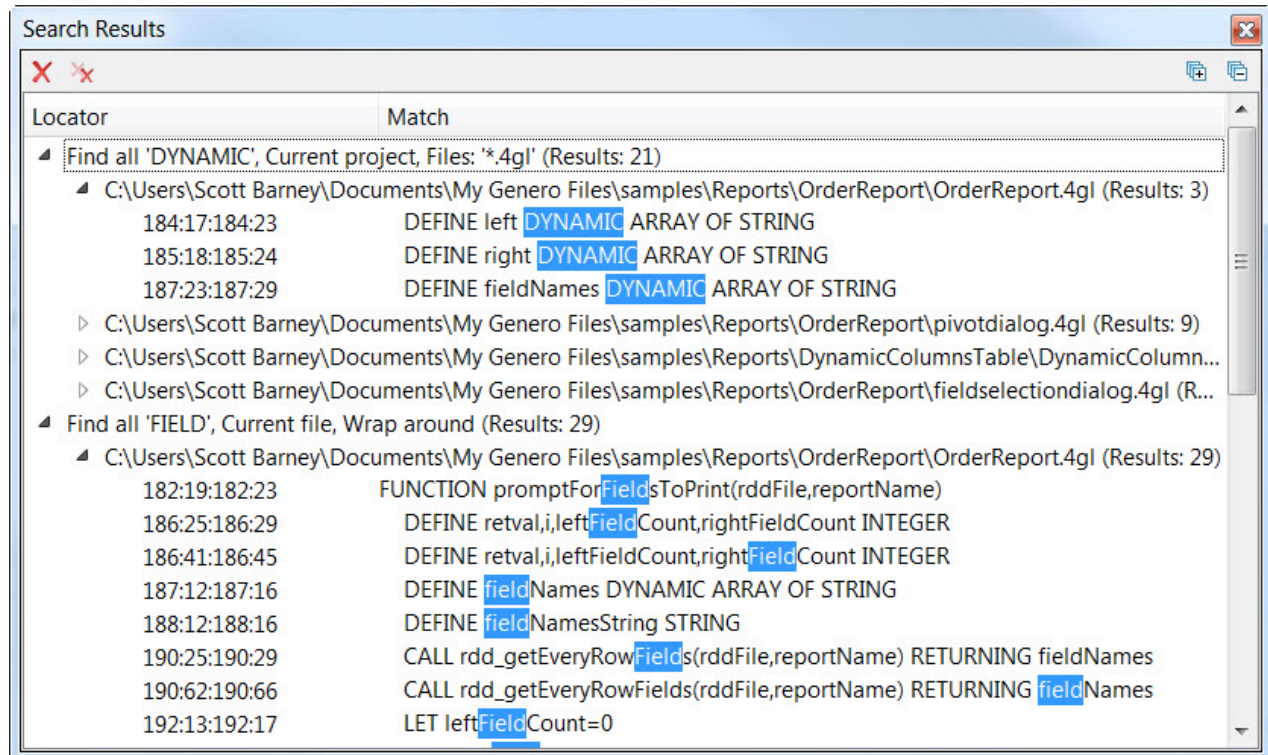


Figure 227: Search results View

The Locator column displays the Search results using a tree view, with three levels:

- The uppermost node provides a summary of the search criteria, as well as the number of results found.
- The first child node represents a file. Each file included in the search gets its own child node. The number of results from within that file are displayed.
- The leaves of the tree view are the individual search results. The format of the locator depends on the searched file. For text files, each result is marked by four numbers: The line where the match starts, the character position where the match starts, the line where the match ends, and the character position where the match ends. For other types of files, it depends on the module used to open the file; the locator is usually an XPath to the model node, or a model node identifier. It will be the same as the locator in the Document Errors view.

The Match column provides the detail of the result in the context of the line from the file, with the search term highlighted.

If more than 10,000 results are found, search is stopped and a message is displayed on the summary line.

Double-click the highlighted text to display the string in its corresponding file.

Tasks view

The Tasks view displays the completion status of current tasks and includes action to abort tasks if needed.

Document Errors view

The Document Errors view displays errors related to a document.

- Select the error number and press the F1 key to display additional information about the error.
- Use [Tools>>Preferences](#), [Genero Studio Preferences](#), [Messages](#) to hide a specific information or warning message.

- View `BUG` or `TODO` notations found in your code. Enter the notation into your code files with `--KEYWORD <message>` where `KEYWORD` is `BUG` or `TODO`. After compiling, put focus in the Project Manager view to see these messages in the Document Errors tab.

Output view

The Output view displays messages related to the output and errors specific to the process being performed.

The **Filter Messages** checkbox shows error messages, warning messages, and/or information messages.

Bookmarks view

The Bookmarks view lists all the bookmarks that have been added to documents in Genero Studio.

You can navigate, add, or remove bookmarks from the Bookmarks view.

The integrated Toolbar and right-click context menu include these options:

Toggle bookmark

Adds or removes a bookmark in the current document, provided the current module supports bookmarks.

Previous / Next bookmark

Activates the previous / next bookmark in the view.

Remove bookmark / Remove all bookmarks

Removes current or all defined bookmarks.

Keyboard Shortcuts

Information about Code Editor keymaps.

- [Cursor movement keymap](#) on page 394
- [Selection keymap](#) on page 395
- [Editing Text keymap](#) on page 396
- [Clipboard keymap](#) on page 397
- [Search and Replace keymap](#) on page 397
- [Buffers and Files keymap](#) on page 397
- [Code Completion keymap](#) on page 398
- [Code Templates keymap](#) on page 398

Note: Genero Studio default accelerator keys may be customized. See [Accelerators](#) in **Tools>>Preferences, User Interface**.

Cursor movement keymap

Table 111: Cursor movement keymap

Command / Action	Keyboard shortcuts
Left one character	Left arrow
Right one character	Right arrow
Left one word	Ctrl + Left arrow
Right one word	Ctrl + Right arrow
Up one line	Up arrow
Down one line	Down arrow
Beginning of line	Home
End of line	End

Command / Action	Keyboard shortcuts
Last index of current line	Alt + End
First index of current line	Alt + Home
Scroll window up one line	Ctrl + Up arrow
Scroll window down one line	Ctrl + Down arrow
Up one screen	Page Up
Down one screen	Page Down
Top of file	Ctrl + Home
Bottom of file	Ctrl + End
Next tab stop	Tab
Previous tab stop	Shift + Tab
Go to line	Ctrl + G
Find matching brace, bracket or parenthesis	Click one brace to highlight matching braces
Smart backspace	Shift + Backspace

Selection keymap

Table 112: Selection keymap

Command / Action	Keyboard shortcuts
Select left one character	Shift + Left arrow
Select right one character	Shift + Right arrow
Select current word	Ctrl + L, W
Select to start of current word	Ctrl + Shift + Left arrow
Select to end of current word	Ctrl + Shift + Right arrow
Select current line	Ctrl + L, L
Select to start of line	Shift + Home
Select to end of line	Shift + End
Select up one line	Shift + Up arrow
Select down one line	Shift + Down arrow
Select to top of window	Ctrl + Shift + Page Up
Select to bottom of window	Ctrl + Shift + Page Down
Select up one screen	Shift + Page Up
Select down one screen	Shift + Page Down
Select to top of file	Ctrl + Shift + Home
Select to bottom of file	Ctrl + Shift + End
Select all	Ctrl + A

Command / Action	Keyboard shortcuts
Square selection	Alt + mouse (left)
Extend square selection to previous character	Alt + Shift + Left arrow
Extend square selection to next character	Alt + Shift + Right arrow
Extend square selection to end of current line	Alt + Shift + End
Extend square selection to start of current line	Alt + Shift + Home

Editing Text keymap

Table 113: Editing Text keymap

Command / Action	Keyboard shortcuts
Toggle Insert / Overstrike mode	Insert
Comment (comments all selected lines)	Ctrl + K, Ctrl + K
Block comment	Ctrl + K, Ctrl + C
Unblock comment	Ctrl + K, Ctrl + U
Delete character/selection	Delete
Delete previous character/selection	Backspace
Delete to end of word (kill-word)	Ctrl + D
Delete to start of word (backward-kill-word)	Ctrl + Backspace
Delete line (kill-line)	Ctrl + Del
Delete to end of line	Ctrl + Shift + Y
Indent block	Tab
Un-indent block	Shift + Tab
Lowercase the currently selected text	Alt + Shift + L
Uppercase the currently selected text	Alt + Shift + U
Toggle case of each character in the selected text	Ctrl + U
Lowercase word	Ctrl + Alt + W
Uppercase word	Ctrl + Alt + Shift + W
Undo	Ctrl + Z Alt + Backspace
Redo	Ctrl + Y Alt + Shift + Backspace
Insert return	Enter Shift + Enter

Clipboard keymap**Table 114: Clipboard keymap**

Command / Action	Keyboard shortcuts
Cut selection	Ctrl + X, Shift + Delete, Cut
Copy selection	Ctrl + C, Ctrl + Insert, Copy
Paste from clipboard	Ctrl + V, Shift + Insert, Paste

Search and Replace keymap**Table 115: Search and Replace keymap**

Command / Action	Keyboard shortcuts
Find	Ctrl + F
Find next	F3
Find previous	Shift + F3
Find in files	Ctrl + Shift + F
Focus back to file	Esc

Buffers and Files keymap**Table 116: Buffers and Files keymap**

Command / Action	Keyboard shortcuts
File New	Ctrl + N
File Open	Ctrl + O
File Print	Ctrl + P
File Save All	Ctrl + Shift + S
File Save	Ctrl + S
File Save As	Ctrl + Alt + S
File Close	Ctrl + F4
Close Genero Studio	Ctrl + Shift + Q
Split window vertically	Ctrl + Alt + V
Split window horizontally	Ctrl + Alt + H
Unsplit window	Ctrl + Alt + Shift + N
Unsplit all	Ctrl + Alt + Shift + A
Preview view	Alt + Shift + P

Code Completion keymap

Table 117: Code Completion keymap

Command / Action	Keyboard shortcuts
Auto Complete	Ctrl + Space

Note: Code Completion is for 4gl files only.

Code Templates keymap

Table 118: Code Templates keymap

Command / Action	Keyboard shortcuts
Code Templates	Ctrl + T

vi Commands List

A listing of vi commands currently implemented in Code Editor.

Table 119: Cursor movement

Keystrokes	Description
h l k j	character left, right; line up, down
b w	word/token left, right
ge e	end of word/token left, right
0 ^ \$	beginning, first, last character of line
nG / :n ngg	line n, default the last, first
B W	space-separated word left, right
gE E	end of space-separated word left, right
H M L	Top, middle, bottom of screen
g0 gm	beginning, middle of screen line
g^ g\$	first, last character of screen line
fc Fc	next, previous occurrence of character c
tc Tc	before next, previous occurrence of c

Table 120: Insertion and Replace (insert mode)

Keystrokes	Description
i a	insert before, after cursor
I A	insert at beginning of first character, end of line
gI (g + Capital I)	insert text in first column/go to first column
o O	open a new line below, above the current line
rc	replace character under cursor with c

Keystrokes	Description
R	replace characters starting at the cursor
cm	change text of movement command m
cc or S	change current line
C	change to the end of line

Table 121: Deletion

Keystrokes	Description
x X	delete character under (right), before (left) cursor
dm	delete text of movement command m
dd D	delete current line, to the end of line
J gJ	join current line with next, without space
:rd	delete range r lines

Table 122: Search and Substitution

Keystrokes	Description
/s# ?s#	search forward, backward for s
n or /#	repeat forward last search
N or ?#	repeat backward last search
# *	search backward, forward for complete word under cursor
g# g*	same, but also find partial matches
gd gD	local, global definition of symbol under cursor
:rs/f /t/x	substitute f by t in range r; x: g -- all occurrences, c -- confirm changes

Table 123: Copying

Keystrokes	Description
ym	yank the text of movement command m
YY/:y or Y	yank current line into register
:ry	yank r range of lines
p P	put register after, before cursor position

Table 124: Undoing, Repeating and Registers

Keystrokes	Description
u U	undo last command, restore last changed line
.	repeat last changes

Keystrokes	Description
.n	repeat last changes with count replaced by n

Table 125: Standard Mode Formatting / Filtering

Keystrokes	Description
~	switch case and advance cursor
g~m gum gUm	switch case, lc, uc on movement m

Table 126: Insert Mode

Keystrokes	Description
esc or ^]	abandon editing # command mode

Table 127: Marks, Motions, and Tags

Keystrokes	Description
mc	mark current position with mark c # [a Z]
'c 'C	go to mark c in current, C in any file

Table 128: Misc IO related commands

Keystrokes	Description
:e f	edit file <i>f</i> , reload current file if no <i>f</i>
:rw f	write range <i>r</i> to file <i>f</i> (current file if no <i>f</i>) (:w, :w f)
:q :q!	quit and confirm, quit and discard changes
:wq or :x or ZZ	write to current file and exit
:r f	insert content of file <i>f</i> below cursor
:n	next file
:p	previous file
:n,kw >> file1	append lines n-k into another file <i>file1</i> .

Table 129: Ranges

Keystrokes	Description
, i	separates two line numbers, set to first line
:n,m	lines n to m
n	an absolute line number n
. \$	the current line, the last line in file
% *	entire file, visual area (exclude)
't	position of mark t
/p/ ?p?	the next, previous line where p matches

Keystrokes	Description
+n -n	+n, #n to the preceding line number

Table 130: Folding

Keystrokes	Description
zo zc zO zC	open, close one fold; recursively
[z]z	move to start, end of current open fold
zj zk	move down/up to start/end of next/previous fold
zm zM	fold more, close all folds
zr zR	fold less, open all folds

Code Analyzer

The *Code Analyzer* reverse engineers existing applications and can generate diagrams to provide an overview of the application.

- [Sequence Diagrams](#) on page 402
- [Dependency Diagrams](#) on page 403

Sequence Diagrams

The Sequence Diagram visually displays the flow of your application logic. It shows how the functions of the application call and/or are called by other functions.

Displaying the Sequence Diagram

To create and display the diagram for a function from an **open 4gl source code file**, right-click the function name to display the context menu option **Open sequence diagram**. To diagram the entire application, right-click the word MAIN in the MAIN program block.

In an open Sequence Diagram, the right-click option **View source** opens the source code module (4gl file) for the function, allowing you to switch back and forth between the source and the diagram.

In the Function calls view of an open Dependency Diagram, the right-click menus for Called Function and Caller Function have an option **Open sequence diagram**.

The Sequence Diagram

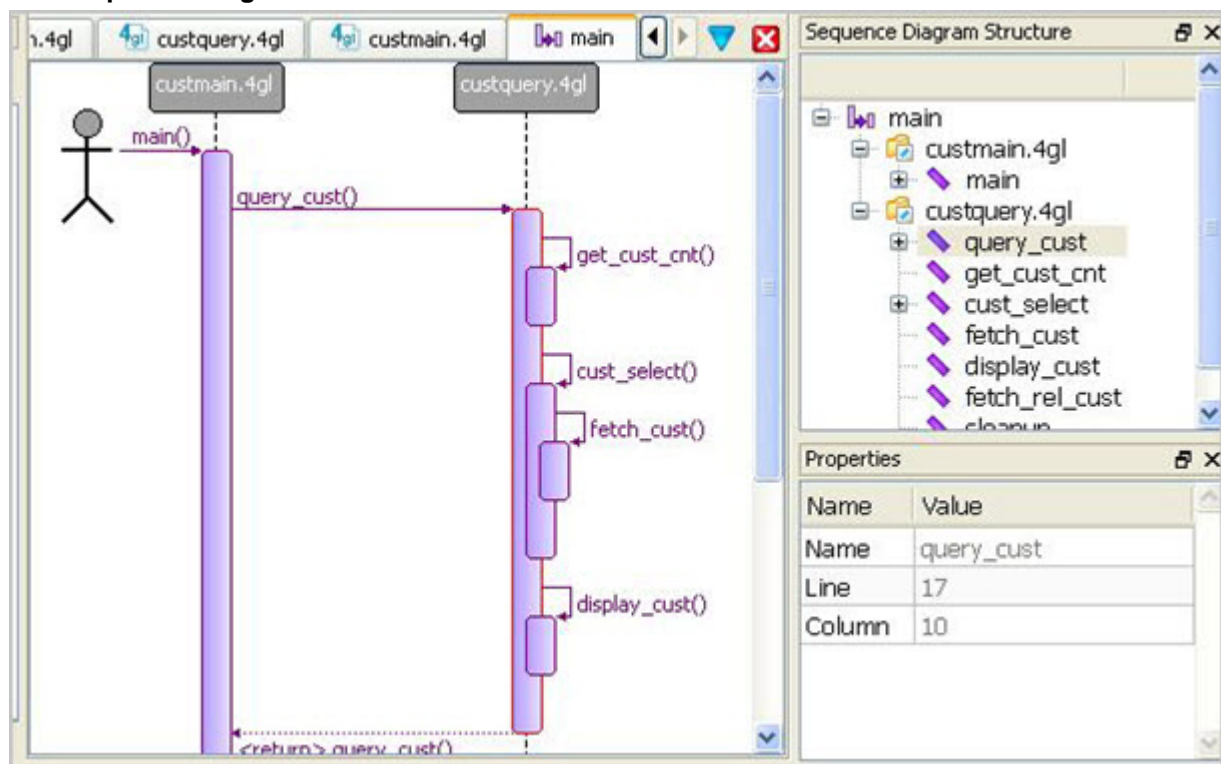


Figure 228: Sequence Diagram

The starting point of the application is indicated by the stick-figure *actor*, the user who interacts with the application.

- Processes that exist simultaneously are represented by *parallel lines* - for a BDL program these lines represent the **source code modules** in your application.
- *Boxes* on the lines represent the **functions** in each module. The sequence is indicated by the order in which the boxes are listed; functions which are called by other functions have their boxes stacked on top. Plus/minus signs on each box allow you to display or hide sub calls.
- *Horizontal arrows* display the interaction between functions, the messages (**calls**) that are exchanged between them, and the order in which the calls occur.

In the example, the user interacts with the **MAIN** program block (function), which calls the **query_cust** function in the `custquery.4gl` source code module. The **query_cust** function calls other functions in that module in the order indicated. For example, it calls the function **cust_select**, which calls **fetch_cust**. The **query_cust** function returns to the MAIN.

Zoom

Use **Ctrl-mouse wheel** to zoom in/zoom out on the diagram.

Structure View

The structure of the program modules is shown in a tree in the Structure view. Use the plus/minus signs to display/hide the functions in a module. Select a function in the Structure view to display its properties in the Properties view. Right-click on a node in the Structure view to see a context sensitive menu of options for that node.

Sub Calls

Right-click on the sequence diagram to display the menu options that will display or hide all the sub calls in a program.

Customize

Use **Tools >> Preferences, Sequence Diagram** to define default maximum number of dependencies; default is 1.

Dependency Diagrams

The Dependency Diagram displays a graphical view of the complex relationships between the various pieces of a project. It shows the components that depend on other components, and/or have components that depend on them.

Displaying the Dependency Diagram

To display a diagram, right-click an Application or Group node, and select **Open Dependency Diagram**.

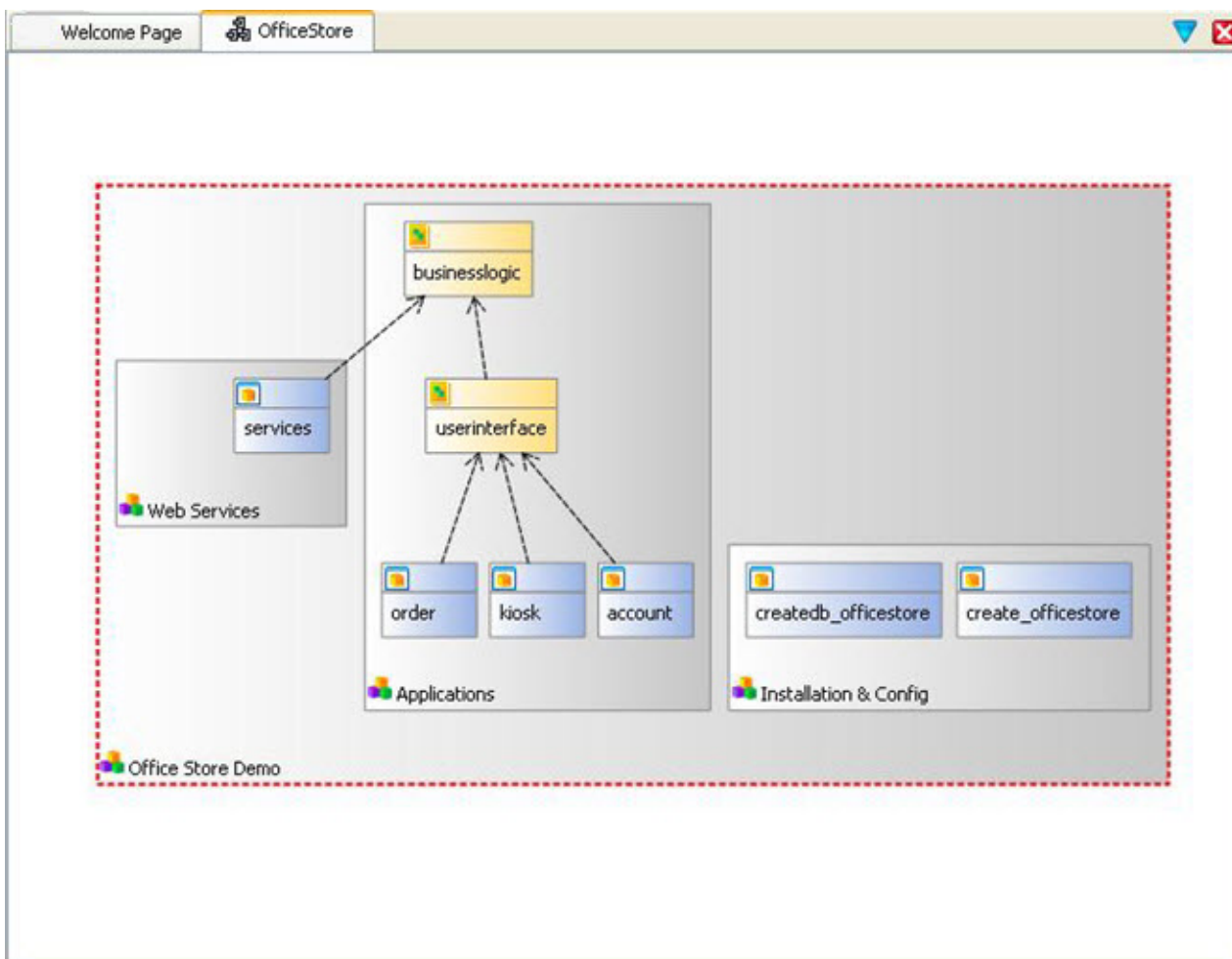


Figure 229: Dependency Diagram

Zoom

Use **Ctrl-mouse wheel** to zoom in/zoom out on the diagram.

Structure and Function Calls Views

The structure of the project is displayed in a tree in the **Structure view**. Select a component in the diagram to display its properties in the **Properties view**. Select the **link** in the diagram between components to display the associated function calls in the **Function calls view**.

Right-click menu options in Function calls view:

- Caller function
 - Symbol definition
 - Open call location
 - Open [sequence diagram](#)
- Called function
 - Symbol definition
 - Open [sequence diagram](#)
- Caller/Called Module
 - View source

Dependency diagram context menu

The Dependency Diagram has a context menu accessible by right-clicking on the diagram or diagram item.

The context menu displays actions applicable to the item selected.

Table 131: Right-click Context Menu

Option	Description
Expand/Collapse	Hide or show subcomponents.
Locate in Project	Bring focus to the component in the project.
Open <i>Project/Application/Library</i> in new tab	Open component in new tab.
Open in new Tab	Open diagram in new tab.
Filter Items ...	The Filter View dialog allows you to hide and show items on a diagram.
Hide selected item / Show all items	Selectively show or hide items.

Form Designer

Form Designer is a visual editor that supports the creation, editing, and layout of Genero forms in Genero Studio.

- [Forms in Genero applications](#) on page 406
- [Quick Start: Creating a first form](#) on page 407
- [Creating the user interface](#) on page 407
- [Form Designer usage](#) on page 440
- [Command-line syntax: gsform](#) on page 442
- [Localizing your form](#) on page 443
- [Form Designer Reference](#) on page 443

Forms in Genero applications

Forms are used in Genero applications to interact with the application's users primarily by allowing for input and display of data.

Built-in functionality automatically detects when a user has entered data into a form, or selected a button or other form item. Frequently the data displayed on the form has been retrieved from a database table or will be used to update the table.

A Genero form is designed in a **Form definition file**. In Form Designer, this form definition file has a `.fd` extension. The form definition file is compiled into an XML file having an extension `.f`. The `.f` is used by the Genero client to display the application form to the user.

Existing `per` forms may be imported and then updated with *Form Designer*.

Program source files (`.gl`) use high-level Genero BDL instructions to open and display the form and to allow users to query a database or make changes in the rows of a database table, for example.

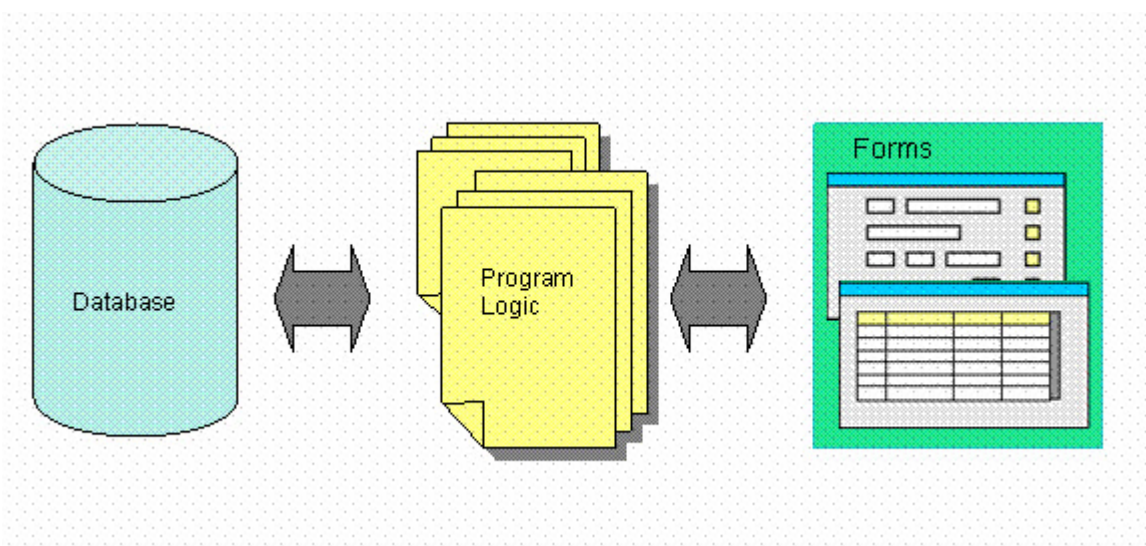


Figure 230: Application flow

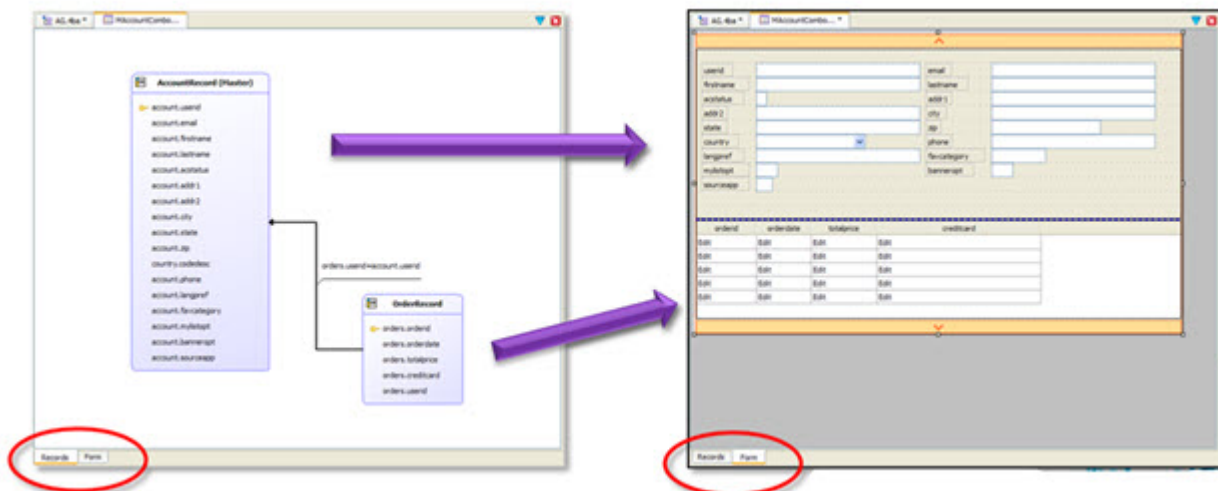


Figure 231: Form Designer diagram

The Form Designer diagram models an application's forms and screen records.

For Genero Mobile apps, see also the *Language and Rendering Guidelines* section in the *Genero Mobile Developer Guide*.

Quick Start: Creating a first form

This topic will guide you through creating a first form in Form Designer.

The easiest way to create a form for data manipulation is from an existing database.

1. [Create a project](#).
2. [Add a meta-schema](#).
3. Select **File >> New Form from Database** and use [the wizard](#) to assist in building the form. Save your form.
4. Enhance form by changing widgets, properties, etc. See [Creating the user interface](#) on page 407.
5. Select **Build >> Preview** to preview your form.
6. Write the code to interact with the form.

Creating the user interface

The Form Designer includes two working areas, one for the Form design and layout and one for the corresponding Record.

For Genero Mobile apps, see also *UI Behavior* in the *Genero Mobile User Guide*.

- [Forms](#) on page 408
- [Containers](#) on page 413
- [Widgets](#) on page 420
- [Web components](#) on page 432
- [Action management \(Toolbars, Topmenus\)](#) on page 434
- [Styles](#) on page 438

Forms

Information about creating and compiling forms.

- [Create a form](#) on page 408
- [Create form from database](#) on page 409
- [Create form with drag and drop](#) on page 409
- [Form item properties](#) on page 410
- [Tab index](#) on page 410
- [Compile a form](#) on page 411
- [Preview a form](#) on page 411

Create a form

You can create new forms in Form Designer.

Before you begin, confirm that you have [added the Meta-schema file \(4db\)](#) to your project so that you can add data fields to your form.

1. Select **File >> New, Genero Files** and choose a form from **Sources**.

Option	Description
Form from Database (4fdm)	Use wizard to select schema, columns and container from which to build new form.
Form (4fd)	Create a blank form (4fd).
Form as Text (per)	Create a blank text form file (per).

2. From the [form design tab](#), [Add a container](#). Form items must be in [Containers](#) on page 413.
3. [Add form items](#). Within a container you can drag-and-drop various form items such as [Widgets](#) on page 420.
4. Each form and form item has properties that control its appearance and behavior. Set properties in the [Properties view](#) on page 448.
5. **Save** the form to the file system and/or project with **File >> Save** or **File >> Save As...**
6. **Preview** the form with **Build >> Preview** to validate that it conforms to Genero rules and see how it will look to users.
7. **Compile** the form with **Build >> Compile File**. You can also compile your form from Project Manager as part of a Build of an entire application.

Add base elements (Containers)

Draw a container from the Container menu onto a form.

All form items must be nested within a container.

1. Open or create a new form.
2. Select a container from the **Container** menu and draw out the container on the form canvas. If you are nesting other objects into the container, draw the container first, then drag the other form elements into the new container. The background of the destination container turns a pale yellow when an object can be nested within it.
3. Set properties on the container as needed. Containers can be resized by selecting one of the corner handles on the container and dragging it to a new size.

Add data elements (Widgets)

Each data elements (widget) added to a form and must be nested with in a container.

Every field on the form is of a specific widget type.

1. Open or create a new form.
2. [Add a container](#).
3. Add widgets.

Option	Description
Widget Menu	Select a widget from the Widget menu and draw out the widget in a container on the form.
Container >> Data Control	Select data fields and a container for the form with the Data Control wizard on page 449. Each field will have a default widget type.
Convert Widget	Select a widget on the form and change the widget type with right-click context menu option Convert Widget .

4. Set properties on the widgets by selecting the widget on the form and setting its properties in the Properties view.

Adding a field to a form adds that field to the corresponding record on the Records tab. Adding a field to the form design outside of any existing containers, results in a new record in the Business Records diagram. You can manage the record and set its properties on the Records tab.

Create form from database

You can create a form from a database using a wizard to select the schema, columns and container from which to build a new form.

Before you begin, confirm that you have [added the meta-schema file](#) to your project.

1. Select **File >> New** and select a category:

Option	Description
Design	Select this option if you are creating a form for use in a Business Application Modeling managed project. The file created will be a <code>4fdm</code> .
Genero Files	Select this option if you are creating a form for use in a standard Genero program. The file created will be a <code>4fd</code> .

2. Select **Form from Database (4fd)** or **CRUD form from Database (4fdm)**. The **New Form from Database** wizard will guide you through setting up the form.
3. [Select the columns](#) from one of more tables to include in the form.
4. [Select the container](#) to use.
5. [Set the relationships](#) between tables if needed.
6. [Change the label and widget](#) used for each field if desired.

Create form with drag and drop

Drag and drop items from the DB Schemas tab to the Form or Records tab.

Before you begin, confirm that you have [added the meta-schema file](#) to your project.

1. Create a new, empty form by selecting **File >> New, Genero Files** and select **Form (.4fd)**.
2. Select the **DB Schemas tab** on page 316 to show available meta-schemas.
3. Select a table, field, or multiple fields (Ctrl-click) from the schema and drag them to the **Form tab** on page 445 or the [Business Record diagram](#) on page 412.
A label and formfield is added for each item dragged to the form. When a table is dragged and dropped, a container is automatically added.

Note:

For [managed forms for generated applications \(4fdm\)](#), if the record is created and the selection contains multiple tables linked with foreign keys, the joins corresponding to the foreign keys are created.

Form item properties

Properties can be set on form items such as containers and widgets to provide information on how the runtime system should display or handle the item.

Properties for a form or form item can be modified by selecting the form item(s) on the form or a node in the Structure view. The properties that are valid for the selected item(s) are displayed.

Some properties are used frequently and are grouped together for easier reading and management. The composition of the groups vary, depending on the selected form item(s).

Default values

Each property has a default value. To reset a default value, use the **Reset value** button next to the property.

Initializers

Some properties include an initializer. The initializer defines how the property's default value is determined. For example, reference information from the database meta-schema (sqltype, notNull, required, uniqueKey, labels, etc.) is inherited to the form by the initializer. Initializers are computed on schema change.

If the property has an initializer option, an **Initializer** button appears to the left of the **Reset value** button and displays the **Initializer** dialog to set or edit the property's initializer value.

When an Initializer property is set or modified, the property name displays in bold and the **Initializer** button changes symbol.

Tab index

The Tab Index indicates the order of the form items for data entry when using the Tab key to move from field to field.

Access the Tab Index by selecting **Widget >> Tab Index**. All items that allow entries are displayed in the form design with an index number specifying the order.

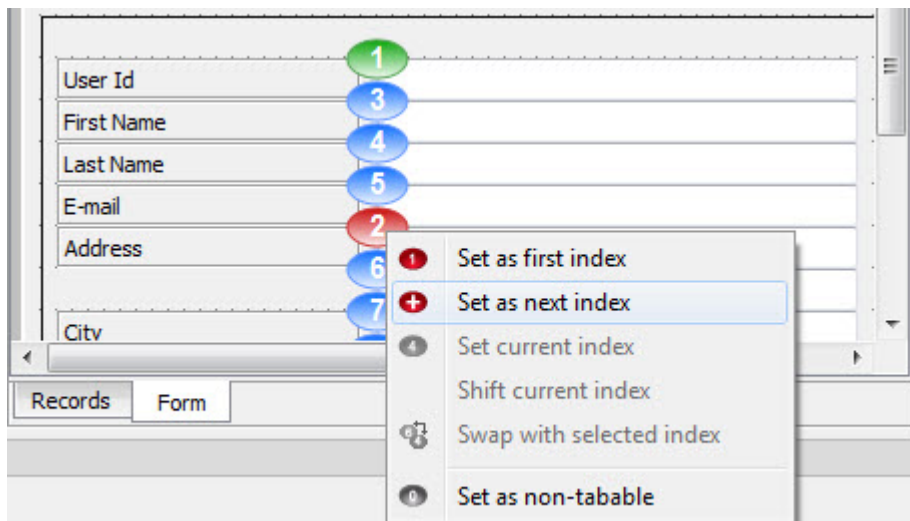


Figure 232: Tab Index

The order can be altered in a variety of ways.

- Double-click a number to change it to 1.
- Click any number to change it to the next number after the last used.
- Right-click and select an option from the context menu.
- Finish the current sequence by pressing the **Esc** key.

- Press the **Esc** key again to exit tab order.

Compile a form

Compiling a form translates it into an XML file with an extension of `42f`.

The compiled form file (`42f`) is used by the Genero client to display the application form to the user. All `42f` files must be included with your application code in distributions.

With the form open, select **Build >> Compile file** from the main menu.

- A form can also be compiled in Project Manager when the form is a part of a Build of an entire application.
- A form can also be compiled at the command-line with `gsform`.

Preview a form

View the form using Genero Desktop Client. The form is automatically validated before the preview process. Only valid forms can be previewed.

Preview a form by opening it and selecting **Build >> Preview**.

Migrate per file to 4fd

A Genero text form (`per`) can be imported and converted into Form Designer form files (`4fd`).

Import an existing Genero `per` file into your project with **File >> Import text form .per...** This action imports and converts the file to a Form Designer form `4fd` file. If the `per` file is already in your project, right-click on the file and choose **Import text form...**

Import an existing Genero `a_per` file and continue to use it in its text form with **Project >> Add files...**

Business records (data sets)

Business records model the data definition, structure and table relationships of the data used in a form, report, and/or web service. Business records are designed and modified in the Business Records diagram.

Forms

Business records for forms model the data definition, structure, and table relationships required to generate the different CRUD operations for the form. When you design a form, a record is automatically created for each container on the form. The Records tab displays the screen record for the form in the Business Records diagram, showing of all the database field names that make up the fields on the form itself.

Reports

Business records for reports (`4rd`) contain the data definition, structure, and table relationships required to generate a `rdd` (Report Data definition file). The `rdd` file is used in conjunction with a Genero report definition (`4rp`) file to automatically generate the reports. A `4rd` file opens in the Business Record diagram.

Services

Business records for web services (`4ws`) contain the data definition, structure, and table relationships required to generate the different CRUD operations for the web service. A `4ws` file opens in the Business Record diagram.

Business records inherit their default information from the meta-schema from which they were created.

Business Record diagram

The Business Record diagram is used to define the data set of the form, report, or web service.

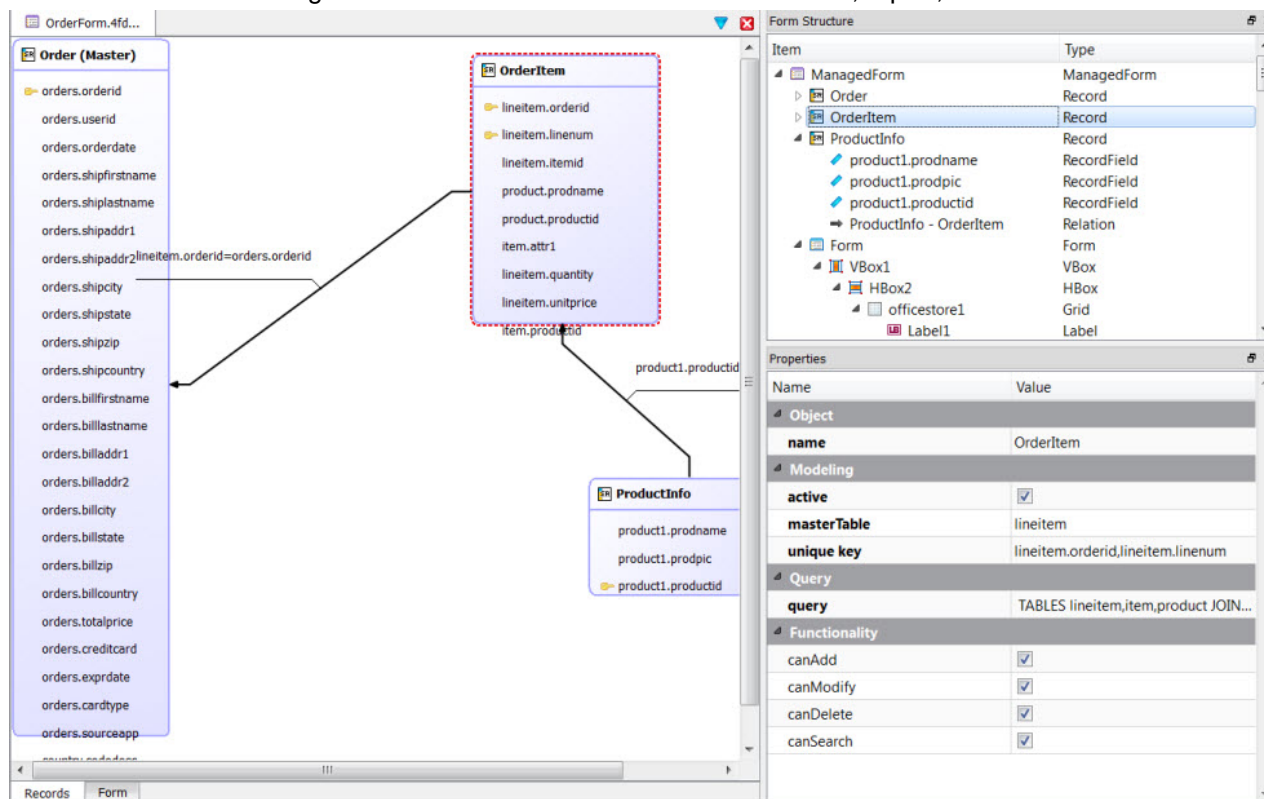


Figure 233: Business Record diagram

Business record properties

Business records have properties that are set in the Properties view.

Table 132: Business Record properties

Property	Description
name	Record name.
active	The <code>active</code> property indicates that the table participates in the application code generation. It is checked by default when a managed form (4fdm) is created. When a table is inactive, it does not provide the modeling features and cannot be or be linked to a master record.
masterTable	A data set can be composed of several database tables. The <i>master table</i> is a table on which CRUD operations will apply. The generator creates the CRUD operations for a given data set only if its unique key is composed by fields of the master table.
unique key	A data set <i>unique key</i> is a field or a list of fields ensuring the uniqueness of data in the data set. It must be defined as a primary key or a secondary key in the database schema.

Property	Description
Functionality	See Enable and disable CRUD logic on page 239. Only for generated forms and web services.
Query	See Joins and Data order on page 453.

Table 133: RecordField properties

Property	Description
name	Record field name.
lookup	See Lookup fields on page 247.
fieldType	Used to specify that its data type is derived from the data types in a database table or that it is Non_Database, indicating that the data type is not derived from a database column. See formFields on page 421.
sqlTabName	The <code>sqlTabName</code> property is the name of the database table for formField form items.
colName	The <code>colName</code> property is the name of the database column.
tableAliasName	Table alias name.
fieldIdRef	Unique reference id of the field in the record.
dataType	See dataType on page 465.
defaultValue	See defaultValue on page 465.

Table 134: Record Relation properties

Property	Description
foreignFields	The foreign key field(s) in the relation.
primaryFields	The primary key field(s) in the relation.

Add or edit a Record, RecordField, or Relation

Records, RecordFields, and Relations are managed in the Business Records diagram.

1. To add a new record, right-click on the background of the Business Records diagram and select **Add Record**.
2. Right-click on a record and select **Add Field** to add a new field to a record. The record and fields are listed in the Structure view.
In Form Designer, new recordFields are also added to the form design in the Form tab.
3. Set the relationship between two records by right-clicking on the foreign field in one record and dragging it to the primary key in the other record. Select the Relation and modify the `foreignFields` and `primaryFields` properties if needed.

Containers

Containers are used to group items on the form. Every form item must be contained within a container. A parent container can also have child containers. If there are multiple child containers, they must be grouped in a horizontal or vertical layout.

- [HBox and VBox - layouting](#) on page 414

- [Grid - positioning](#) on page 415
- [ScrollGrid - positioning](#) on page 415
- [Group - grouping](#) on page 416
- [Folder page - stacking](#) on page 417
- [Table - organizing](#) on page 417
- [Tree - hierarchy](#) on page 418
- [HRec - aligning fields](#) on page 419
- [Matrix](#) on page 420
- [Data Control](#) on page 420

Note: Some containers are not supported on mobile platforms.

HBox and VBox - layouting

HBox and VBox containers are used to layout the containers nested within them either horizontally from left to right (HBox) or vertically from top to bottom (VBox).

Containers are packed in the HBox or VBox container in the order in which they appear in the structure. By combining HBox and VBox layouts for various containers, you can align the containers of your form in any way you choose.

Example

Figure 234: HBox Layout Container

Example

Figure 235: VBox Layout Container

Properties

[name](#) on page 475, [posX](#), [posY](#) on page 477, [gridHeight](#), [gridWidth](#) on page 470

Grid - positioning

The Grid container declares a formatted text block defining the dimensions and the positions of the form elements for a unique-record presentation.

With Grid, you can specify the position of labels, formFields for data entry or additional interactive objects such as buttons. Grids have no visual representation when the form is displayed.

Example

Figure 236: Grid Container

Properties

[name](#) on page 475, [tag](#) on page 480, [style](#) on page 480, [posX](#), [posY](#) on page 477, [gridHeight](#), [gridWidth](#) on page 470, [hidden](#) on page 470, [fontPitch](#) on page 467, [comment](#) on page 464, [localizedStr](#)

ScrollGrid - positioning

The ScrollGrid container declares a formatted text block defining the dimensions and the positions of the form elements for a multi-record presentation.

Example

ScrollGrid is similar to the Grid container, except that you can repeat the screen elements on several "row-templates", in order to design a multiple-record view that appears with a vertical scrollbar. A ScrollGrid may be a container of, or contained within, both Grids and ScrollGrids.

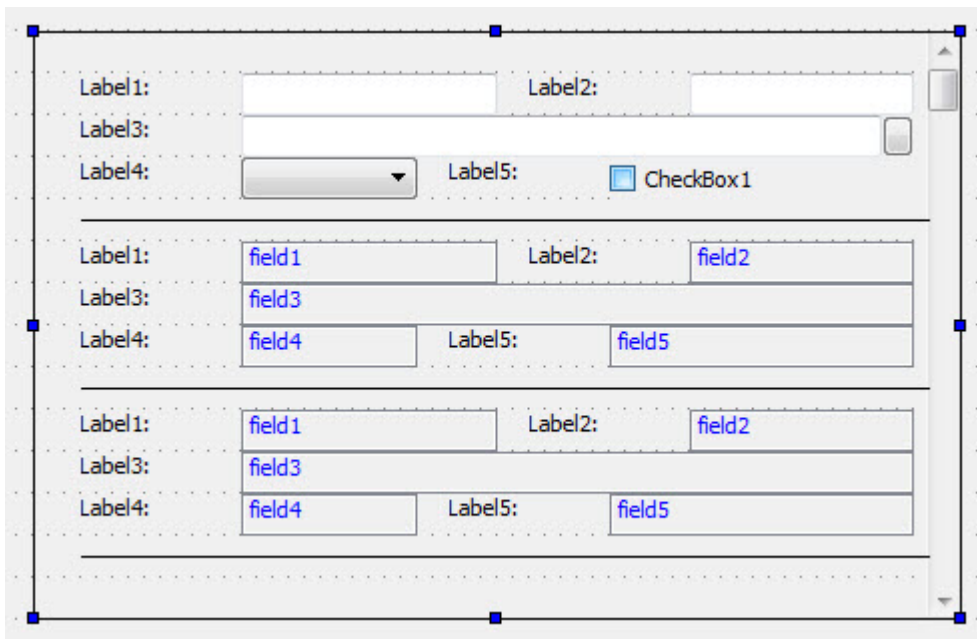


Figure 237: ScrollGrid Container

Properties

[name](#) on page 475, [tag](#) on page 480, [style](#) on page 480, [posX](#), [posY](#) on page 477, [gridHeight](#), [gridWidth](#) on page 470, [hidden](#) on page 470, [fontPitch](#) on page 467, [comment](#) on page 464, [gridChildrenInParent](#) on page 470, [doubleClick](#) on page 466, [localizedStr](#)

Group - grouping

A Group container can be used to display a titled box (usually called a *groupbox*) around contained elements.

To display a groupbox around a set of fields, nest your Grid or other container within a Group container.

Example



Figure 238: Group Container

Properties

[name](#) on page 475, [tag](#) on page 480, [style](#) on page 480, [posX](#), [posY](#) on page 477, [gridHeight](#), [gridWidth](#) on page 470, [hidden](#) on page 470, [fontPitch](#) on page 467, [comment](#) on page 464, [gridChildrenInParent](#) on page 470, [doubleClick](#) on page 466, [text](#) on page 481

Folder page - stacking

A Folder container is used to display children (Pages) inside a Folder tab.

The Folder container manages the Page containers displayed as tabs. You can add or remove and order the Pages using the right-click contextual menu on the Folder or Page object on the form.

Example

The screenshot shows a form with three tabs: "Billing", "Shipping", and "Credit Card". The "Billing" tab is selected and active. Inside the Billing tab, there is a form with the following fields: "First Name", "Last Name", "Address", "City", "State", "Zip Code", and "Country". The "Country" field is a dropdown menu.

Figure 239: Folder and Pages

Properties

[action](#) on page 462, [name](#) on page 475, [tag](#) on page 480, [style](#) on page 480, [posX](#), [posY](#) on page 477, [gridHeight](#), [gridWidth](#) on page 470, [hidden](#) on page 470, [fontPitch](#) on page 467, [comment](#) on page 464, [localizedStr](#)

Usage

The **text** property defines the label of the folder page. The **image** property can be used to specify which image to use as an icon.

If needed, you can use the **action** property to bind an action to a folder Page. When the Page is selected, the program gets the corresponding action event.

To bring a folder page to the top, in your source code, use `NEXT FIELD` to one of the active fields of the page (The `NEXT FIELD field-name` instruction gives the focus to the specified field.) or use the `ui.Form.ensureFieldVisible()` method if the fields are disabled/unused or the `ui.Form.ensureElementVisible()` method if the page does not contain focusable elements.

With the **tabIndex** property of a field in a Page, you can define which field gets the focus when a Page is selected.

Table - organizing

The Table container defines the presentation of a list of records, bound to a (also called a *screen array*).

Each record is displayed on a separate line. The screen record appears on the Records tab.

Example

Order Id	Order Date	Customer Type	Total Price
Edit	Edit	RadioGroup	Edit
Edit	Edit	RadioGroup	Edit
Edit	Edit	RadioGroup	Edit
Edit	Edit	RadioGroup	Edit
Edit	Edit	RadioGroup	Edit
			Total:

Figure 240: Table Container

Properties

[name](#) on page 475, [tag](#) on page 480, [style](#) on page 480, [posX](#), [posY](#) on page 477, [gridHeight](#), [gridWidth](#) on page 470, [hidden](#) on page 470, [fontPitch](#) on page 467, [comment](#) on page 464, [localizedStr](#), [wantFixedPageSize](#) on page 484, [unSortableColumns](#) on page 482, [unSizableColumns](#) on page 482, [unMovableColumns](#) on page 481, [unHidableColumns](#) on page 481, [doubleClick](#) on page 466, [totalRows](#) on page 481, [rowHeight](#) on page 478

Usage

Drag and drop the columns to change the initial order.

To add or delete columns or edit a column title, select the column title and right-click to display a menu of options.

The screen record definition on the Records tab must have exactly the same columns as the Table container. However, the order of the screen record fields can be different from the column order, to match the program array elements, for example when the database table defines the columns (`DEFINE LIKE`) in a different order as the form table.

By default, the current row in a Table is highlighted in display mode, but it is not highlighted in input mode. You can set decoration attributes of a table with a style.

If the **aggregate** property is set for one of the columns in the table, a summary line will be displayed.

Tree - hierarchy

The Tree container defines the presentation of a list of ordered records in a tree-view widget.

Example

Edit2	Edit3
Edit	Edit
Edit	Edit
Edit	Edit
Edit	Edit
Edit	Edit

Figure 241: Tree Container

Properties

[name](#) on page 475, [tag](#) on page 480, [style](#) on page 480, [posX](#), [posY](#) on page 477, [gridHeight](#), [gridWidth](#) on page 470, [hidden](#) on page 470, [fontPitch](#) on page 467, [comment](#) on page 464, [localizedStr](#), [wantFixedPageSize](#) on page 484, [unSortableColumns](#) on page 482, [unSizableColumns](#) on page 482, [unMovableColumns](#) on page 481, [unHidableColumns](#) on page 481, [doubleClick](#) on page 466, [totalRows](#) on page 481, [rowHeight](#) on page 478, [parentIdColumn](#) on page 476, [idColumn](#) on page 471, [expandedColumn](#) on page 467, [isNodeColumn](#) on page 474, [imageCollapsed](#) on page 472, [imageLeaf](#) on page 473

Usage

Tree views are very similar to regular table containers; before reading further about tree views, you should be familiar with Table containers. The Tree container allows you to specify the layout of a graphical tree widget, displaying data in a parent-child relationship. Plus and minus icons allow the user to expand and contract a branch of the tree.

You specify the layout for the tree in Form Designer; the tree content is provided in your BDL program.

A Tree is made of rows and columns, very similar to a Table container, with the exception that the rows in a tree can be nested.

The Tree container is created by default with two Edit columns, and two Phantom columns that do not display on the screen. The Phantom columns of the Tree specify the id and the parent id of each row, enabling nesting. The other columns display data rows in a normal format without nesting, as in a table.

Properties of the tree container specify the `idColumn` and `parentIdColumn`. Additional properties specify images to be used for the expand/contract icons and leaf icons.

The rows in the tree are automatically defined in your form as a screen array. In your BDL program, define a corresponding dynamic array that matches the screen array, using the screen record in the Form to determine column names and their order. Your program must populate the rows of the container at runtime.

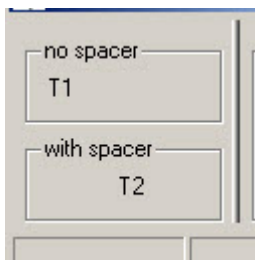
See the topic on Tree Views in the *Genero Business Development Language User Guide* for more details about tree-view programming in Genero.

HRec - aligning fields

An HRec item is a special container that uses spacers to align widgets in a form.

Spacers are one or more blanks defining an invisible element that expands automatically. Without spacers, blank areas are ignored and the resulting form may not display as you expect. Draw out the HRec container and then drag and drop form items into it. Once you have placed form items into the HRec container you can right-click on the form item to add additional spacer objects to the right or left of the form item. The spacer objects force the spacing between the form items within the HRec container.

HRec Container



Properties

[name](#) on page 475, [posX](#), [posY](#) on page 477, [gridHeight](#), [gridWidth](#) on page 470

Matrix

A Matrix container manages a screen array, usually a repetitive array of fields in the screen layout, each containing identical groups of screen fields.

Each row of a screen array is a screen record. Right-click on a field to **Convert to Matrix** and set its Matrix properties. The Structure View marks matrix fields with a preceding + sign.

Properties

Table 135: Matrix Properties

Property	Description
repeat	Determines if field repeats. Default is checked (TRUE).
columnCount	Number of columns of repeated fields.
rowCount	Number of rows of repeated fields. Minimum value is 1. Both columnCount and rowCount cannot be 1, otherwise the field would be a standard formField and not a repeated field.
stepX	Number of cells (horizontally) between the repeated fields. (Relevant only if columnCount > 1.)
stepY	Number of cells (vertically) between the repeated fields. (Relevant only if rowCount > 1.)

Data Control

Although this item appears on the Container menu, it is actually a wizard that allows you to create a new form or add multiple fields to the form, based on database table columns.

Widgets

Widgets are designed for data handling, action triggering, or decoration.

- [Button](#) on page 421
- [ButtonEdit](#) on page 421
- [Canvas](#) on page 422
- [CheckBox](#) on page 422
- [ComboBox](#) on page 423
- [DateEdit](#) on page 424
- [Edit](#) on page 425
- [Field](#) on page 426
- [HLine](#) on page 426
- [Image](#) on page 426
- [Label](#) on page 427
- [Phantom](#) on page 428
- [ProgressBar](#) on page 428
- [RadioGroup](#) on page 429
- [Slider](#) on page 430
- [SpinEdit](#) on page 430
- [TimeEdit](#) on page 432
- [TextEdit](#) on page 431

formFields

A formField is a type of form object that can be used to display data or take input.

A formField is presented to the user through a widget; the most commonly used widget, Edit, defines a simple line edit box that allows the user to enter a value directly into the formField. Other widgets, such as the ComboBox and Checkbox, present the data contained in the field in a user-friendly way. For example, a ComboBox defines a dropdown box of values, allowing the user to select from a list of valid values for the underlying formField.

In the Properties View, the fieldType property of a formField may be set to specify that its data type is derived from the data types in a database table or that it is Non_Database, indicating that the data type is not derived from a database column.

If the data type is to be derived from a database table, the databaseName property of the form must be set to a database for which there is a database meta-schema file listed in **the Db Schemas tab**. The [sqlTabName](#) and [colName](#) properties of the formField must be selected from the tables and columns in the schema file. The [dataType](#) property will be automatically filled based on the data type for the table column in the schema file.

A form has built-in validation to insure that the value entered into a formField is compatible with the declared [dataType](#) in the form definition file. Additional validation routines can be specified in your BDL program.

The values entered into formFields are stored in variables in the Genero program, which may be used in the program in any way. A common use is to provide values for SQL statements that update a database.

Button

The Button widget defines a push-button with a label or picture that can trigger an action defined in your BDL program code.

When a button is clicked, the program code defined for the action is triggered.

Example

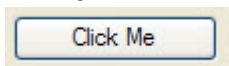


Figure 242: Button

Properties

[action](#) on page 462, [comment](#) on page 464, [fontPitch](#) on page 467, [hidden](#) on page 470, [image](#) on page 471, [sample](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [tabIndex](#) on page 480, [text](#) on page 481, [tag](#) on page 480

ButtonEdit

A ButtonEdit widget defines a line edit box with a button on the right side.

Purpose

The ButtonEdit editable box is a [formField](#), which can be associated with a database column by changing the [fieldType](#) property to TABLE_COLUMN, and specifying the [sqltabName](#) and [colName](#) properties. The button can trigger an action defined in your BDL program code. Frequently this action is designed to open a window that displays a list of values for the user to choose from.

Example

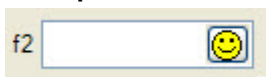


Figure 243: ButtonEdit

Properties

[action](#) on page 462, [autoNext](#) on page 462, [case](#) on page 463, [century](#) on page 463, [color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [Display Like](#) on page 466, [case](#) on page 463, [fontPitch](#) on page 467, [hidden](#) on page 470, [format](#) on page 467, [image](#) on page 471, [include](#) on page 473, [invisible](#) on page 473, [justify](#) on page 474, [notNull](#) on page 476, [noEntry](#) on page 475, [picture](#) on page 476, [reverse](#) on page 478, [sample](#) on page 478, [scroll](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [required](#) on page 477, [tag](#) on page 480, [tabIndex](#) on page 480, [Validate Like](#) on page 482, [verify](#) on page 483

Canvas

A Canvas widget defines an area reserved for drawing.

Example

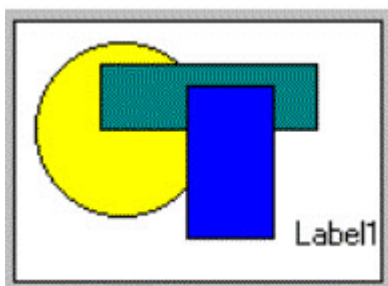


Figure 244: Canvas

See the topic on Canvas in the *Genero Business Development Language User Guide* for more information.

Properties

[comment](#) on page 464, [hidden](#) on page 470, [tag](#) on page 480

CheckBox

The CheckBox widget defines a boolean entry with a box and a text label.

Example



Figure 245: CheckBox

Properties

[color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [fontPitch](#) on page 467, [color](#) on page 464, [include](#) on page 473, [justify](#) on page 474, [notNull](#) on page 476, [noEntry](#) on page 475, [required](#) on page 477, [sample](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [tag](#) on page 480, [tabIndex](#) on page 480, [text](#) on page 481, [Validate Like](#) on page 482, [valueChecked](#) on page 482, [valueUnchecked](#) on page 483

Usage

The [text](#) on page 481 attribute defines the label to be displayed near the check box.

The box shows a checkmark when the form field contains the value defined in the [valueChecked](#) attribute (for example: "Y"), and shows no checkmark if the field value is equal to the value defined by the [valueUnchecked](#) attribute (for example: "N"). If you do not specify the [valueChecked](#) or [valueUnchecked](#) attributes, they respectively default to TRUE (integer 1) and FALSE (integer 0).

By default, during an INPUT, a CheckBox field can have three states:

- Grayed (NULL value)
- Checked ([valueChecked](#) value)
- Unchecked ([valueUnchecked](#) value)

If the field is declared as `notNull`, the initial state can be grayed if the default value is NULL; once the user has changed the state of the CheckBox field, it switches only between checked and unchecked states.

During an CONSTRUCT, a CheckBox field always has three possible states (even if the field is `notNull`), to let the user clear the search condition:

- Grayed (No search condition)
- Checked (Condition column = [valueChecked](#) value)
- Unchecked (Condition column = [valueUnchecked](#) value)

ComboBox

A ComboBox is a data handling widget that defines a drop-down list of values, allowing the user to select a value.

Purpose

The ComboBox underlying formField can be associated with a database column by changing the `fieldType` property to `TABLE_COLUMN`, and specifying the `sqlTabName` and `colName` properties.

Example

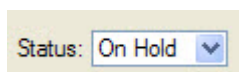


Figure 246: Combobox

Properties

[case](#) on page 463, [color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [case](#) on page 463, [fontPitch](#) on page 467, [hidden](#) on page 470, [include](#) on page 473, [initializer](#) on page 473, [items](#) on page 474, [justify](#) on page 474, [notNull](#) on page 476, [noEntry](#) on page 475, [queryEditable](#) on page 477, [required](#) on page 477, [sample](#) on page 478, [scroll](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [tag](#) on page 480, [tabIndex](#) on page 480, [Validate Like](#) on page 482

Usage

The values of the drop-down list are defined by the `items` property. You can define a simple list of values like ("A", "B", "C", "D", . . .) or you can define a list of key/label combinations like in ((1, "Paris"), (2, "Madrid"), (3, "London")). In the latter, the labels (i.e. the city names) will be displayed according to the key value (the city number) hold by the field.

The `initializer` property allows you to define an initialization function for the `COMBOBOX`. This function will be invoked at runtime when the form is loaded, to fill the item list dynamically with database records, for example. It is recommended that you use the `tag` property, so you can identify in the program the kind of ComboBox form item to be initialized.

If neither `items` nor `initializer` properties are specified, the form compiler automatically fills the list of items with the values of the `include` property, when specified. However, the item list will not automatically be populated with include range values (i.e. values defined using the `TO` keyword). The `include` property can be specified directly in the form or indirectly in the schema files.

During an INPUT, a ComboBox field value can only be one of the values specified in the [items](#) property. During an CONSTRUCT, a ComboBox field gets an additional 'empty' item (even if the field is [notNull](#)), to let the user clear the search condition.

If one of the items is explicitly defined with NULL and the [notNull](#) property is omitted; In INPUT, selecting the corresponding combobox list item sets the field value to null. In CONSTRUCT, selecting the list item corresponding to null will be equivalent to the = query operator, which will generate a "colname is null" SQL condition.

During a CONSTRUCT, a ComboBox is not editable by default: The end-user is forced to set one of the values of the list as defined by the [items](#) property, or set the 'empty' item. The [queryEditable](#) property can be used to force the ComboBox to be editable during a CONSTRUCT instruction, in order to allow free search criterion input such as "A*". If [queryEditable](#) is used and the [items](#) are defined with key/label combinations, the text entered by the user will be automatically searched in the list of items. If a label corresponds, the key will be used in the SQL criterion, otherwise the text entered by the user will be used. For example, if the items are defined as ((1, "Paris"), (2, "Madrid"), (3, "London")), and the user enters "Paris" in the field, the item (1, "Paris") will match and will be generate "colname = 1". If the user enters ">2", the text does not match any item so it will be used as is and generate the SQL "colname > 2". Users may enter values like "Par*", but in this case the runtime system will raise an error because this criterion does is not valid for the numeric data type of the field. To avoid end-user confusion, a ComboBox defined with key/label combinations should not use the [queryEditable](#) property.

Some front-ends support different presentation options which can be controlled by a [style](#) property. You can for example enable the first item to be selected when pressing keys.

Context Menu

The values in the list of items can be managed (add, delete) by right-clicking on the ComboBox on the form in Form Designer and selecting **Edit Items**. Edit the values in the columns of the dialog:

- **Name** - the value to be stored in the underlying formField represented by the ComboBox, for example, "W".
- **Text** - the value in the list to be displayed to the user, for example "West Region".
- **Localize string** - Specify whether the text is a localized string (true/false).

DateEdit

A DateEdit is a data handling widget that defines a line edit box with a button that opens a calendar window, allowing the user to select a date value

Purpose

The dateEdit underlying formField can be associated with a database column by changing the [fieldType](#) property to TABLE_COLUMN, and specifying the sqltabName and columnName properties.

Example

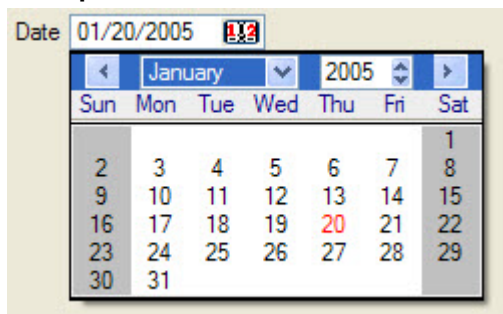


Figure 247: DateEdit

Properties

[autoNext](#) on page 462, [century](#) on page 463, [color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [fontPitch](#) on page 467, [format](#) on page 467, [hidden](#) on page 470, [include](#) on page 473, [justify](#) on page 474, [notNull](#) on page 476, [noEntry](#) on page 475, [required](#) on page 477, [sample](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [tag](#) on page 480, [tabIndex](#) on page 480, [Validate Like](#) on page 482

Usage

Some front-ends support different presentation options which can be controlled by a [style](#) on page 480 attribute. For example, you can change the first day of the week or the icon of the button.

DateTimeEdit

The DateTimeEdit widget defines a date-time editor widget.

Purpose

The DateTimeEdit form item type allows the user to edit date-time values with a specific widget for date-time input. A DateTimeEdit field typically provides a calendar and clock widget, to let the end user pick a date and time from it.

The display and input precision (time part with or without seconds) of the DateTimeEdit widget depends from the front-end. On some platforms, native date-time editors do not handle the seconds. Further, some front-ends (especially on mobile devices) deny data types different from DATETIME DAY TO {MINUTE|SECOND}.

To store DateTimeEdit field values, consider using the appropriate data type according to the target front-end (DATETIME YEAR TO SECOND or DATETIME YEAR TO MINUTE).

Important: If the front-end does not support the data type used for the DateTimeEdit field, the runtime system will raise an error and stop the program. Consider testing your application with all type of front-ends.

On some front-end platforms, the native widget used for DateTimeEdit fields allows only pure date-time value input, and therefore cannot be used with a CONSTRUCT instruction, where it must be possible to enter search filters like ">= 2014-01-23 11:00".

Properties

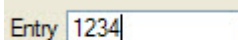
[autoNext](#) on page 462, [century](#) on page 463, [color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [fontPitch](#) on page 467, [format](#) on page 467, [hidden](#) on page 470, [include](#) on page 473, [justify](#) on page 474, [notNull](#) on page 476, [noEntry](#) on page 475, [required](#) on page 477, [sample](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [tag](#) on page 480, [tabIndex](#) on page 480, [Validate Like](#) on page 482

Edit

The Edit item type defines a simple line-edit field for data input or display.

The Edit item type is a data handling widget for an underlying [formfield](#). The Edit formField can be associated with a database column by changing the [fieldType](#) property to TABLE_COLUMN, and specifying the sqltabName and columnName properties.

Example



Entry

Figure 248: Edit item type

Attributes

[autoNext](#) on page 462, [century](#) on page 463, [color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [Display Like](#) on page 466, [fontPitch](#) on page 467, [format](#) on page 467, [hidden](#) on page 470, [include](#) on page 473, [justify](#) on page 474, [notNull](#) on page 476, [noEntry](#) on page 475, [required](#) on page 477, [sample](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [tag](#) on page 480, [tabIndex](#) on page 480, [Validate Like](#) on page 482, [verify](#) on page 483

Field

The Field item type defines a generic form field for data input or display that is defined in database schema files.

Example

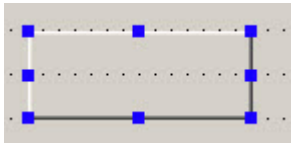


Figure 249: Field

Properties

[comment](#) on page 464, [defaultValue](#) on page 465, [fontPitch](#) on page 467, [hidden](#) on page 470, [notNull](#) on page 476, [noEntry](#) on page 475, [required](#) on page 477, [sample](#) on page 478, [style](#) on page 480, [sizePolicy](#) on page 478, [tag](#) on page 480, [tabIndex](#) on page 480

Usage

This item type defines a generic form field for data input or display. The real item type (i.e. the widget) and the attributes must be defined in the database schema files.

The definition of the form field is determined by the `.val` database schema file, based on the *field-name* (table.column). The item type (Edit, ComboBox, etc) is defined by the ITEMTYPE attribute in the `.val` schema file.

By using this form field specification, you can centralize the definition of form fields in the database schema file, to enforce reusability. You can, for example, specify that the "order.state" database column is a [ComboBox](#) on page 423, with a list of [items](#) on page 474, as if the field was defined directly in the form.

It is also possible to use the properties defined in the database schema files with other Form Item types.

The properties defined directly in the form take precedence over the properties defined in the database schema files.

The database schema files can be edited manually or by using the [Meta-schema Manager](#) on page 288.

HLine

The HLine item type appears in the form as a horizontal line.

Example

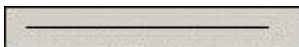


Figure 250: HLine

Image

Image items define areas where a picture file can be displayed.

Images can be either static or dynamic formFields.

Example



Figure 251: Image

Properties

[autoScale](#) on page 463, [comment](#) on page 464, [hidden](#) on page 470, [style](#) on page 480, [stretch](#) on page 480, [tag](#) on page 480

Static Image only: [image](#) on page 471

Image Field only: [color](#) on page 464, [colorCondition](#) on page 464, [fontPitch](#) on page 467, [justify](#) on page 474, [sizePolicy](#) on page 478, [sample](#) on page 478

Usage

A **Static Image** displays an image that has its source file defined by the [image](#) property. This type of image item must be used to display text that does not change often, such as background pictures or logos. The item is not a formField. The image file can be changed from the BDL program by using the API provided to manipulate the user interface (see the *Genero Business Development Language User Guide* topic **Dynamic User Interface** for more details). It is not possible to change the image with a DISPLAY TO instruction.

A **formField Image** is a widget that gets the image file based on the underlying [formField](#). The value of the formField is the image file specified with a URL. The formField can be associated with a database column by changing the [fieldType](#) on page 467 property to TABLE_COLUMN, and specifying the sqltabName and colName properties. This type of image item must be used to display values that change often during program execution, like database information. The picture can be changed from the BDL program by using the DISPLAY TO instruction to set the value of the field.

Label

The Label item type defines a simple text area to display a read-only value.

Labels can be either static or dynamic formFields.

Example

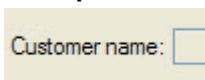


Figure 252: Label

Properties

[color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [fontPitch](#) on page 467, [hidden](#) on page 470, [imageColumn](#) on page 472, [justify](#) on page 474, [reverse](#) on page 478, [sample](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [tag](#) on page 480

FormField Label only: [format](#) on page 467, [sample](#) on page 478

Static Label only: [text](#) on page 481

Table Column only: [unHidable](#) on page 481, [unMovable](#) on page 481, [unSizable](#) on page 481, [unSortable](#) on page 482, [title](#) on page 481

Usage

A **formField Label** item is used to display values that change often during program execution, like database information. The text of the label is defined by the value of the corresponding form field. The text can be changed from the BDL program by using the DISPLAY TO instruction to set the value of the field, or within a list by using a DISPLAY ARRAY. This kind of Form Item does not allow data entry; it is only used to display values. The text automatically changes when the values in the table column change.

Some front-ends support different presentation options which can be controlled by a [style](#) attribute. You can for example change the text format to render HTML content.

A **Static Label** item is used to display text that does not change often, like field descriptions. The text of the label is defined by the [text](#) attribute; the item is not a form field. Double-click the widget, or right-click and select **Edit Text**, to edit the text property. The text can be also changed from the BDL program by using the API provided to manipulate the user interface (see Dynamic User Interface for more details). It is not possible to change the text with a DISPLAY TO instruction. This kind of item is not affected by instructions such as CLEAR FORM. Static labels display only character text values, and therefore do not follow any justification rule as form field labels.

Phantom

A Phantom field can be used to specify a formField that is listed in a screen-record, but does not have to be displayed in the form.

Usage

Phantom fields are never displayed to the user, although they can be used by dialog instructions of BDL programs. If you want to implement a screen-array with all the columns of a database table defined in the [schema file](#), but you don't want to display all the columns in the [table](#), you must use Phantom fields. With the screen-array matching the database table, you can easily write program code to fetch all columns into an array defined with a LIKE clause.

Phantom fields can be based on database columns defined in a schema file or as NON_DATABASE field.

Phantom fields are used to store the id and parent id of the nodes in a [tree](#) object.

To add a Phantom field to your form, right-click on the form object and select **Add Phantom** from the context menu.

Phantom field data is never send to the front-ends. Therefore, you can use a phantom field to store critical data that must not go out of the application server.

ProgressBar

A ProgressBar is a data handling widget that can indicate the current progress of an operation.

Example



Figure 253: ProgressBar

Properties

[color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [fontPitch](#) on page 467, [hidden](#) on page 470, [justify](#) on page 474, [sample](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [tag](#) on page 480, [valueMin](#) on page 483, [valueMax](#) on page 483

Usage

A ProgressBar item does not allow data entry; it is only used to display integer values.

The position of the progress bar is defined by the value of the corresponding form field. The value can be changed from the BDL program by using the DISPLAY TO instruction to set the value of the field.

The `valueMin` and `valueMax` properties define respectively the lower and upper integer limit of the progress information. Any value outside this range will not be displayed. Default values are `VALUEMIN=0` and `VALUEMAX=100`.

Some front-ends support different presentation options which can be controlled by a `style` property. For example, you could display a percentage.

This widget has to be used with a SMALLINT or INTEGER variable, larger types like BIGINT or DECIMAL are not supported.

RadioGroup

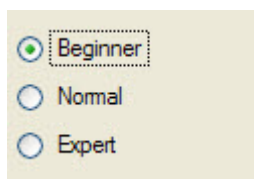
A RadioGroup data handling widget presents a set of radio buttons from which the user may select a value.

Purpose

The underlying formField for the RadioGroup can be associated with a database column by changing the `fieldType` property to TABLE_COLUMN, and specifying the `sqltabName` and `colName` properties. Adding/deleting values for the radio buttons is managed from the contextual menu.

Example

Figure 254: RadioGroup



Property

[color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [fontPitch](#) on page 467, [hidden](#) on page 470, [include](#) on page 473, [items](#) on page 474, [justify](#) on page 474, [notNull](#) on page 476, [noEntry](#) on page 475, [orientation](#) on page 476, [required](#) on page 477, [sample](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [tag](#) on page 480, [tabIndex](#) on page 480, [Validate Like](#) on page 482

Usage

This item type defines a set of radio buttons where each button is associated with a value defined in the [items](#) on page 474 property.

The text associated with each value will be used as the label of the corresponding radio button.

If the [items](#) property is not specified, the form compiler automatically fills the list of items with the values of the [include](#) property, when specified. However, the item list will not automatically be populated with include range values (i.e. values defined using the TO keyword). The item property can be specified directly in the form or indirectly in the schema files.

During an INPUT, a RadioGroup field value can only be one of the values specified in the [items](#) property. During an CONSTRUCT, a RadioGroup field allows to uncheck all items (even if the field is [notNull](#)), to let the user clear the search condition.

If one of the items is explicitly defined with NULL and the [notNull](#) property is omitted; In INPUT, selecting the corresponding radio button sets the field value to null. In CONSTRUCT, selecting the radio button corresponding to null will be equivalent to the = query operator, which will generate a "colname is null" SQL condition.

Use the [orientation](#) property to define if the radio group must be displayed vertically or horizontally.

Some front-ends support different presentation options which can be controlled by a [style](#) property. For example, you can define what item has to be selected first when pressing keys.

Slider

The Slider item type defines a horizontal or vertical slider.

Example

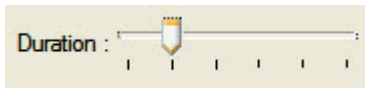


Figure 255: Slider

Properties

[color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [fontPitch](#) on page 467, [hidden](#) on page 470, [include](#) on page 473, [justify](#) on page 474, [orientation](#) on page 476, [sample](#) on page 478, [sizePolicy](#) on page 478, [step](#) on page 480, [style](#) on page 480, [tabIndex](#) on page 480, [tag](#) on page 480, [Validate Like](#) on page 482, [valueMin](#) on page 483, [valueMax](#) on page 483

Usage

The slider item type is a data handling widget for the underlying formField. The formField can be associated with a database column by changing the [fieldType](#) property to TABLE_COLUMN, and specifying the sqltabName and colName properties.

This item type defines a classic widget for controlling a bounded value. It lets the user move a slider along a horizontal or vertical groove and translates the slider's position into a value within the legal range.

The [valueMin](#) and [valueMax](#) properties define respectively the lower and upper integer limit of the slider information. Any value outside this range will not be displayed; the step between two marks is defined by the [step](#) property. The [orientation](#) property defines whether the Slider is displayed vertically or horizontally. If valuemIn and/or valuemax are not specified, they default respectively to 0 (zero) and 5.

This widget has to be used with a SMALLINT or INTEGER variable, larger types like BIGINT or DECIMAL are not supported.

Note: This widget is not designed for CONSTRUCT, as you can only select one value.

SpinEdit

The SpinEdit item type defines a spin box widget.

Example

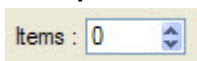


Figure 256: SpinEdit

Properties

[autoNext](#) on page 462, [color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [fontPitch](#) on page 467, [hidden](#) on page 470, [include](#) on page 473, [justify](#) on page 474, [notNull](#) on page 476, [noEntry](#) on page 475, [required](#) on page 477, [sample](#) on page 478, [sizePolicy](#) on page 478, [step](#) on page 480, [style](#) on page 480, [tabIndex](#) on page 480, [tag](#) on page 480, [Validate Like](#) on page 482, [valueMin](#) on page 483, [valueMax](#) on page 483

Usage

This item type allows the user to choose a value either by clicking the up/down buttons to increase/decrease the value currently displayed, or by typing the value directly into the spin box.

The step between two values is defined by the [step](#) attribute.

The [valueMin](#) and [valueMax](#) attributes define respectively the lower and upper integer limit of the spin-edit range. There is no default min or max value for the `SPINEDIT` widget.

This widget has to be used with a `SMALLINT` or `INTEGER` variable, larger types like `BIGINT` or `DECIMAL` are not supported.

This widget is not designed for `CONSTRUCT`, as you can only enter an integer value.

TextEdit

The `TextEdit` item type defines a multi-line edit field for data input or display.

Example

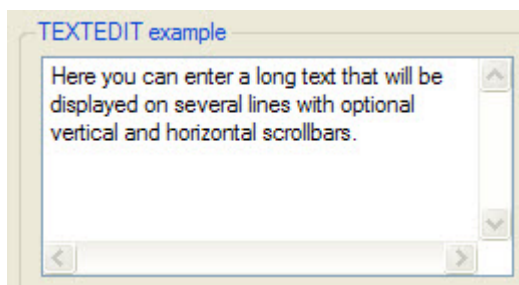


Figure 257: TextEdit

Properties

[case](#) on page 463, [color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [fontPitch](#) on page 467, [hidden](#) on page 470, [include](#) on page 473, [justify](#) on page 474, [noEntry](#) on page 475, [notNull](#) on page 476, [required](#) on page 477, [sample](#) on page 478, [scrollbars](#) on page 478, [sizePolicy](#) on page 478, [stretch](#) on page 480, [style](#) on page 480, [tabIndex](#) on page 480, [tag](#) on page 480, [Validate Like](#) on page 482, [wantTabs](#) on page 484, [wantNoReturns](#) on page 484

Usage

This kind of form field allows the user to enter a long text on multiple lines.

By default, when the focus is in a `TextEdit` field, the `TAB` key moves to the next field, while the `RETURN` key adds a newline (`ASCII 10`) character in the text. To control the user input when the `TAB` and `RETURN` keys are pressed, you can specify the [wantTabs](#) and [wantNoReturns](#) properties. When you specify [wantTabs](#), the `TAB` key is consumed by the `TextEdit` field, and a `TAB` character is added to the text. The user can still jump out of the field with the `Shift-TAB` combination. When you specify [wantNoReturns](#), the `RETURN` key is not consumed by the `TextEdit` field, and the action corresponding to the `RETURN` key is triggered. The user can still enter a newline character with `Shift-RETURN` or `Control-RETURN`.

You can use the [scrollbars](#) property to define vertical and/or horizontal scrollbars. By default, this attribute is set to `Vertical`. The [stretch](#) property can be used to force the `TextEdit` field to stretch when the parent container is re-sized. Values can be `NONE`, `X`, `Y` or `BOTH`. By default, this attribute is set to `NONE`. Note that using either the `SCROLLBARS` or the `STRETCH` attribute will automatically set the `SCROLL` attribute. For more details about size limitation, see the [scroll](#) property.

Some front-ends support different text formats which can be controlled by a [style](#) property. You can for example display and input `HTML` content in a `TextEdit`.

Since Genero 2.20, a TextEdit can also be used to edit rich text format. Depending on the front-end, different formatting options are available (bold, font size, and so on) and can be controlled using either an integrated toolbox or via local actions. In this case, the value of the field will be an HTML representation of the text and its decoration.

Note: Each front-end uses its underlying technology to provide this feature and the html representation may vary between front-ends (GDC uses Qt QTextEdit, GWC/Ajax and GWC/HTML uses the browser capabilities, GWC/SL uses Silverlight capabilities). They are most of the time compatible but not every time, and the html representation may change depending on the version (Qt upgrade for GDC, Silverlight upgrade for GWC/SL, browser update for GWC/Ajax or HTML5).

Note: When using rich text, FGL_DIALOG_SETCURSOR() and FGL_DIALOG_SETSELECTION() functions must be called carefully. The rich text format, having a corresponding cursor position / selection between displayed text and html representation, may make it difficult.

TimeEdit

The TimeEdit item type defines a time editor widget.

Example

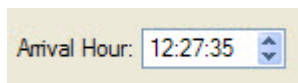


Figure 258: TimeEdit item type

Properties

[autoNext](#) on page 462, [color](#) on page 464, [colorCondition](#) on page 464, [comment](#) on page 464, [defaultValue](#) on page 465, [fontPitch](#) on page 467, [hidden](#) on page 470, [include](#) on page 473, [justify](#) on page 474, [noEntry](#) on page 475, [notNull](#) on page 476, [required](#) on page 477, [sample](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [tabIndex](#) on page 480, [tag](#) on page 480, [Validate Like](#) on page 482

Usage

This item type allows the user to edit times by using the keyboard or the arrow keys to increase/decrease time values.

With this widget, the user can only enter a DATETIME HOUR TO SECOND value.

This widget is not designed for CONSTRUCT, as you can only enter time.

Web components

Web Components are usually complex widgets displaying detailed information on the screen, such as charts, graphs, or calendars.

- [Web component widget](#) on page 432
- [Add a WebComponent to a form](#) on page 433
- [WebComponent setup \(advanced\)](#) on page 433

Web component widget

The WebComponent item type defines a generic form field that can receive an external widget.

Properties

[color](#) on page 464, [colorCondition](#) on page 464, [componentType](#) on page 465, [comment](#) on page 464, [defaultValue](#) on page 465, [fontPitch](#) on page 467, [hidden](#) on page 470, [include](#) on page 473, [justify](#) on page 474, [noEntry](#) on page 475, [notNull](#) on page 476, properties, [required](#) on page

477, [scrollbars](#) on page 478, [sizePolicy](#) on page 478, [style](#) on page 480, [stretch](#) on page 480, [tabIndex](#) on page 480, [tag](#) on page 480, [Validate Like](#) on page 482

Usage

The WebComponent item type defines a form field which can be implemented with a plug-in mechanism on the front-end side.

You must define the type of the widget with the [componentType](#) property. This property is mandatory to identify the external widget that will be used for this field.

The [scrollbars](#) and [stretch](#) properties can be used to define the behavior of the widget regarding sizing.

The [properties](#) property is typically used to define properties that are specific to a given WebComponent. For example, a chart component might have properties to define x-axis and y-axis labels.

The value of a WebComponent field is usually (XML) formatted, and holds the data that will be rendered by the external widget through the JavaScript™ shell.

Note: In order for a WebComponent to be listed in the [componentType](#) list, it must be described in an XML file contained in the directory that you specified in *GSTWCDIR*.

See the topic on WebComponents in the *Genero Business Development Language User Guide* for more information.

Add a WebComponent to a form

Each WebComponent widget is described in an XML description file having the extension **.wcsettings**.

All the description files must be stored in a single directory. This directory must also contain any icon files to be displayed on the form design page for the WebComponent. Specify the directory using the [GSTWCDIR](#) on page 144 environment variable in a Web Component environment set. See [Environment sets](#) on page 140. Once you have set this directory, you may add the widgets to your form design documents.

1. Select **Widget>>WebComponent** from the Genero Studio menu and add the widget to the form.
2. Choose the specific WebComponent from the combobox for the [componentType](#) property.

When you add a [WebComponent](#) to your form design, a standard image is displayed in the form, indicating that the object is a WebComponent. The [componentType](#) property in the WebComponent section of the [Properties View](#) allows you to select the specific WebComponent that you wish to add. Once you select the component, the properties specific to that component are also listed in the Properties view; if an icon associated with this component has been created for the form design page, it will be displayed.

WebComponent setup (advanced)

To make Web Components available for Genero Studio form designers, the description of the Web Component must be described in an XML file having the extension **.wcsettings**.

A separate XML file must be created to describe each type of WebComponent, **storecalendar.wcsettings** for example. The [componentType](#) property in the Properties View is a combobox that displays the names of the available WebComponent widgets ("storecalendar" for example).

You can display an icon on the form design document when a web component is selected; otherwise the default icon is displayed. The name of the icon file must be the same as that of the wcsettings file, for example, **storecalendar.wcsettings** and **storecalendar.jpg**.

The form designer must specify the directory that contains the wcsettings and icon files; see [Add a WebComponent to a form](#) on page 433.

WebComponent description files

Example file **chart.wcsettings**:

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<WebComponent>
  <DynamicProperty name="type" type="TEXT" label="type"
    description="type of chart"
    initialValue="" />
  <DynamicProperty name="caption" type="TEXT" label="caption"
    description="caption"
    initialValue="" />
  <DynamicProperty name="subcaption" type="TEXT" label="subcaption"
    description="subcaption" initialValue="" />
  <DynamicProperty name="xaxisname" type="TEXT" label="X label"
    description="label of X axis" initialValue="" />
  <DynamicProperty name="yaxisname" type="TEXT" label="Y label"
    description="label of Y axis" initialValue="" />
  <DynamicProperty name="numberPrefix" type="TEXT" label="numberPrefix"
    description="number prefix" initialValue="" />
  <DynamicProperty name="labels" type="STRINGLIST" label="labels"
    description="labels" initialValue="" />
  <DynamicProperty name="values" type="STRINGLIST" label="value"
    description="values"
    initialValue="" />
</WebComponent>

```

The XML schema description for the .wcsettings file is contained in an XSD file named **wcsettings.xsd**, located in the Genero Studio installation directory **conf/schema**.

Action management (Toolbars, Topmenus)

Information about adding Toolbars and Topmenus to forms.

- [Action views](#) on page 434
- [Action defaults](#) on page 434
- [Topmenus](#) on page 435
- [Toolbars](#) on page 437

Action views

Action views on a form such as Toolbars, Topmenus, and buttons, trigger actions defined in your program code.

The value of the name or the [action](#) property, must exactly match the name of an action in an ON ACTION clause of an interactive statement, such as MENU.

For example, a [ButtonEdit](#) on the form with the action property value set to "search" would trigger the action "**search**" in this BDL program code, when the MENU statement is active and the user clicks the button.

```

MENU
  ON ACTION search
    CALL find_customer()
  . . .

```

Action defaults

Action defaults can be used to define the appearance of action views on the form (buttons, menu items, and Toolbar items, for example).

For more information see "Understanding action defaults" in the *BDL User Guide*.

Add action defaults

Action defaults can be added to a form.

1. Open form.
2. Right-click the form object in the Structure view and select **Add Action Default List** from the context menu.

The form structure now includes an **ActionDefaultList** node.

3. Right-click on the **ActionDefaultList** node in the form structure view and select **Add Action Default**. Repeat to add additional ActionDefaults.
4. For each ActionDefault, specify the properties associated with text, images, accelerators, and comments. These properties will be automatically applied to any [action views](#) on the form that have the same name as the property of the ActionDefault in the list.

Import action defaults

An existing action defaults file (4ad) can be imported into a form.

1. Open form.
2. Right-click the form object in the Structure view and select **Import Action Defaults** from the context menu.
3. Navigate the file system to find the action default file (4ad) to be used. Select **Open** to import the file into your form.
The form structure now includes an **ActionDefaultList** node with **ActionDefault** nodes for each action default.
4. These properties will be automatically applied to any [action views](#) on the form that have the same name as the property of the ActionDefault in the list.
Additional ActionDefaults added to the list will be used for the form, but will not effect the 4ad file imported.

Topmenus

A Topmenu presents a pull-down menu on the form associated with actions defined by the current interactive instruction in the BDL code.

When a Topmenu command is selected by the user, the BDL program triggers the action to which the Topmenu command is bound.

A Topmenu can be defined directly in the form or in a resource file with the extension 4tm.

See the topic on Topmenus in the *Genero Business Development Language User Guide* for more information.

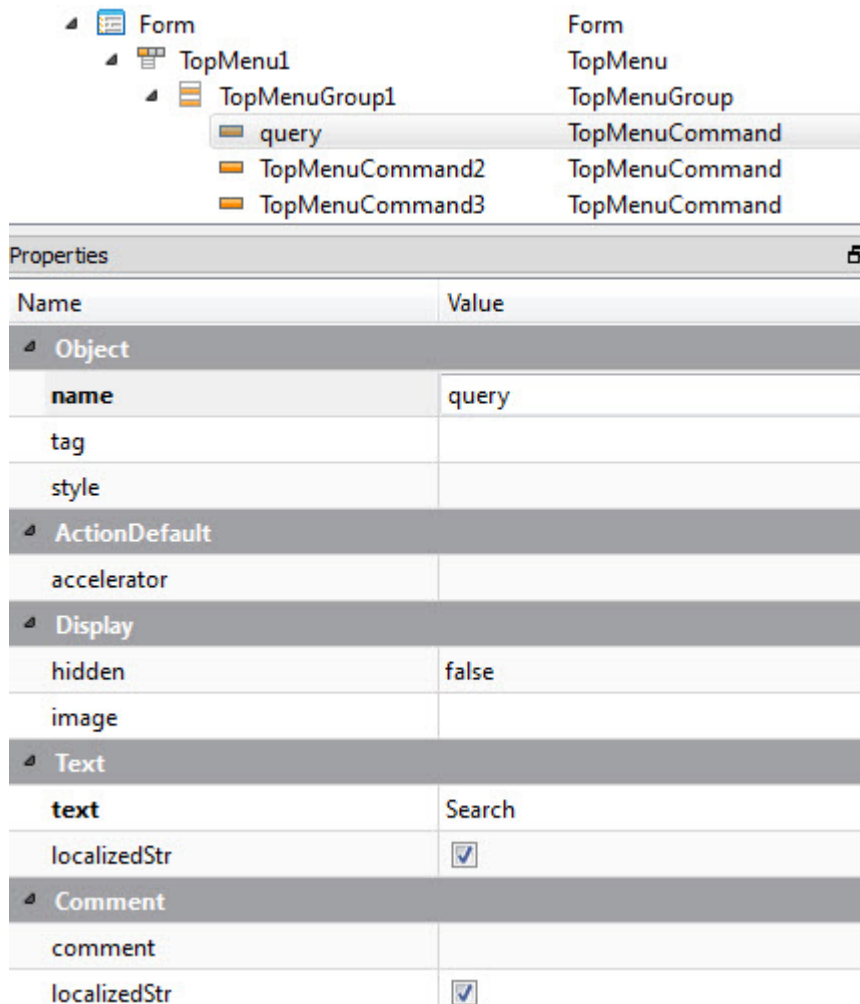


Figure 259: Form Structure tab showing a Topmenu

Add a Topmenu

A Topmenu can be added to a form.

Adding a Topmenu at the form level will make the Topmenu available only to that form. To create a Topmenu the can be used for all forms or as needed, create a `4tm` file and add it to your project.

1. Open form.
2. Right-click the form object in the Structure view and select **Add Top Menu** from the context menu. The form structure now includes an **Topmenu** node and the Topmenu structure is added to the form design.
3. Build your Topmenu:
 - Build your Topmenu directly in the form design by using the icons at the top of the form.

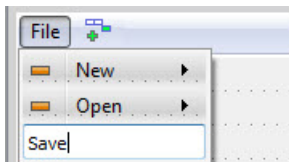


Figure 260: Topmenu editor

- Or, right-click on the **Topmenu** node in the form structure view and add the elements of the Topmenu.

Option	Description
Add Group	Adds a TopMenuGroup, the text and parent node for the pull-down menu group.
Add Command	Adds a TopMenuCommand and specifies the action the menu option must be bound to. Note: The TopMenuCommands's <code>name</code> property must exactly match the <code>name</code> of an action specified in your BDL program.
Add Separator	Adds a TopMenuSeparator, an optional horizontal line to be displayed on the Topmenu, visually separating some of the commands.

4. Set properties for each of the elements in the Topmenu.

Import a Topmenu

An existing Topmenu (4tm) can be imported into a form.

1. Open form.
2. Right-click the form object in the Structure view and select **Import Top Menu** from the context menu.
3. Navigate the file system to find the Topmenu file (4tm) to be used. Select **Open** to import the file into your form.
The form structure now includes a **Topmenu** node with nodes for each of the elements defined in the Topmenu. The Topmenu also appears at the top of the form in the form design window.
4. Set properties for each of the elements in the Topmenu as needed.
Changes made to the Topmenu will be used for the form, but will not effect the 4tm file imported.

Toolbars

A Toolbar presents buttons on the form associated with actions defined by the current interactive instruction in the BDL code.

When a Toolbar button is selected by the user, the program triggers the action to which the Toolbar button is bound.

The Toolbar object appears in the Form Structure View, but it does not appear in the design window. It implements a Toolbar on the form that is displayed to the user.

A Toolbar can be defined directly in a form or in a resource file with the extension 4tb.

See the topic on Toolbars in the *Genero Business Development Language User Guide* for more information.

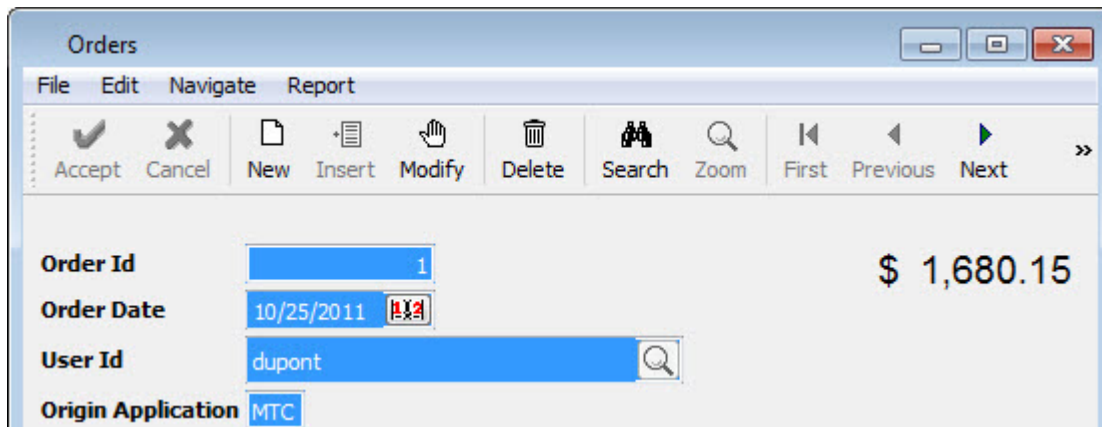


Figure 261: Form with Toolbar displayed using the Genero Desktop Client

Add a Toolbar

A Toolbar can be added to a form.

Adding a Toolbar at the form level will make the Toolbar available only to that form. To create a Toolbar the can be used for all forms or as needed, create a `4tb` file and add it to your project and program.

1. Open form.
2. Right-click the form object in the Structure view and select **Add Tool Bar** from the context menu. The form structure now includes an **ToolBar** node.
3. Right-click on the **ToolBar** node in the form structure view and add the elements of the Toolbar.

Option	Description
Add Item	Adds a <code>ToolBarItem</code> node and specifies the action the Toolbar button must be bound to. Note: The <code>ToolBarItem</code> 's name property must exactly match the name of an action specified in your BDL program.
Add Separator	Adds a <code>ToolBarSeparator</code> , an optional vertical line to be displayed on the Toolbar, visually separating some of the commands.

4. Set properties for each of the elements in the Toolbar.

Import a Toolbar

An existing Toolbar (`4tb`) can be imported into a form.

1. Open form.
2. Right-click the form object in the Structure view and select **Import Tool Bar** from the context menu.
3. Navigate the file system to find the Toolbar file (`4tb`) to be used. Select **Open** to import the file into your form. The form structure now includes a **ToolBar** node with nodes for each of the elements defined in the Toolbar.
4. Set properties for each of the elements in the Toolbar as needed. Changes made to the Toolbar will be used for the form, but will not effect the `4tb` file imported.

Styles

Styles allow you to centralize properties related to the appearance of user interface elements.

Typical style properties define font properties and foreground and background colors. Some style properties will be specific to a given class of widgets (like the first day of week in a `DATEEDIT`).

Styles can be defined directly in the form or in a resource file having an extension of **4st**, which must be distributed with other runtime files.

Apply a style to a form item

Styles defined in a style file (`4st`) can be applied to forms and form items for form previews and style dialogs..

Before you begin, you need a style file defined in order to apply styles to your form items. Styles are defined in a resource file with the extension (`4st`).

1. Set the `styleFile` property for the form. To explicitly set the style file to use:
 - a) Select the form node in the Structure view to display its properties.
 - b) Select the `styleFile` property.
 - c) Click the `...` button to browse on your system for the style file that you want to use for your form.

If no style file is specified in the `styleFile` property, and a `default.4st` file is found in the path specified by the `FGLRESOURCEPATH` environment variable, the `default.4st` file is used for form previews and style dialogs.

If you add the `default.4st` file to a library in your project, and a dependency to this library exists, then the directory is automatically added to `FGLRESOURCEPATH`.

If no `default.4st` file is found in `FGLRESOURCEPATH`, the `default.4st` file from `<FGLDIR>/lib` is used.

This provides flexibility in the use of style files. You can:

- Specify the style file in the form file.
- Add a `default.4st` style file to a group node, and set a dependency to the form application node.
- Put a `default.4st` in a directory, and add the path to the `FGLRESOURCEPATH` environment variable of an active environment set.
- Automatically use the `FGLDIR/lib/default.4st` file.

Important: The `styleFile` attribute is only used by the Form Designer when previewing the form and for Form Designer-related style dialogs. To apply a style file with your Genero application, you must load the style file using the `ui.Interface.loadStyles` method. See the *Genero Business Development Language User Guide* for more information regarding the use of style files by Genero applications.

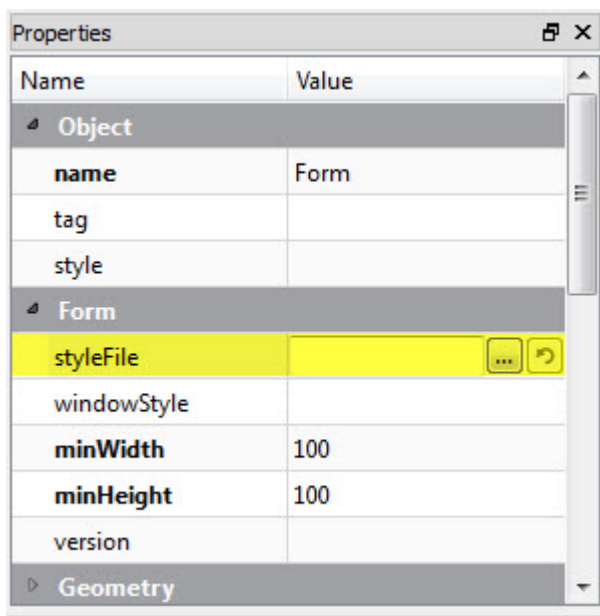


Figure 262: styleFile property in Properties view

2. Select the form item or items for which you want to apply a style.
3. In the properties view, select the style property, and click the ... button to display the **Style Selection dialog**. Select a style or styles for the form item(s). Select **OK**.
Selected styles are listed, separated by a space, in the style property for the form item.

Create a style file

You can create a style file (`.4st`) for your forms.

Select **File >> New >>**

Option

Design, Resources >> Style (.4st)

Description

Creates a style file using the `dbapp.4st` template as a basis.

Option	Description
Genero Files, Resources >> Style (.4st)	Creates a style file using the default .4st template as a basis.

Form Designer usage

Information about designing a form with Form Designer.

- [Drawing](#) on page 440
- [Selecting, moving, resizing](#) on page 440
- [Aligning](#) on page 441
- [Transforming](#) on page 441

Drawing

Containers and widgets can be drawn onto the form.

Using the Toolbar of containers and widgets, you can add one or more container or widget of a specific type onto the form.

Single-click the Toolbar icon

The item is selected, and one occurrence can be placed on the form.

Double-click the Toolbar icon

The item remains selected until explicitly deselected by clicking in the main window, another icon, or the select arrow. This allows you to draw several occurrences of the same item without re-selecting it.

Nesting

All items on the form are in a hierarchy identified in the Structure view. All form items must be nested within a container. Some containers can be nested within other containers. For example, a table container could be nested within a folder container. To place a widget within a container on the form design, draw the widget in the container, or drag the drawn widget into a container. The parent container changes color indicating that the widget can be nested within it. The structure view will show the widget within the parent container. You can also use the structure view to change the parenting of containers to containers and widgets within containers by dragging and dropping the selected item into its new location in the structure.

Selecting, moving, resizing

Form items can be selected individually or in groups, moved, and resized.

Selecting

Select an item

Use the mouse to select an item. A selected item on the form design is displayed with handles around it.

Double-click an item

Double-clicking on an item executes the default action, which depends on the widget. For example, double-clicking on a button label or group opens the **Edit text** dialog. Double-clicking on a combobox or radiogroup, opens the **Edit items** dialog.

Multiple selection

To select multiple items, select an item and then hold down the Ctrl key while selecting additional items.

Square selection

While holding down the Shift key, drag a rectangle touching elements to select.

Select all

Right click on the form and select **Select all Form Fields**. This selects all form fields within the current container.

Ctrl-A (select all) selects all elements on the current tab (Form or Record).

Moving and resizing

Selected items can be moved with the mouse or by using the keyboard arrows (left, right, up, down). This changes the value of the item's posX and/or posY attributes.

Selected items can be re-sized by holding down the Shift key while using the keyboard arrows. This changes the value of the item's gridWidth and/or gridHeight attributes.

Aligning

- [Align widgets](#) on page 441
- [Layout containers](#) on page 441

Align widgets

Widgets within the same container may be aligned.

The parent container size is limited to the form's maximum width and height.

Align items by using the mouse to select. Use Ctrl-mouse click to select multiple items. Select an option from the **Alignment** menu.

Layout containers

If a parent container has more than one child container, the child containers must be grouped and aligned using layouts.

Select the containers to be grouped. Use the Layout menu to group them. The form must be large enough to contain both the child containers in their desired position, as well as the layout container.

Horizontal Layout

Aligns the containers horizontally in a horizontal layout container.

Vertical Layout

Aligns the containers vertically in a vertical layout container.

Break Layout

Removes layout, ungrouping the containers; if you have grouped containers in a layout, you must break the layout in order to move the containers.

Transforming

Form items can be transformed from one type to another.

- [Convert a widget](#) on page 441
- [Convert a container](#) on page 442
- [Convert to matrix](#) on page 442

Convert a widget

Right-click on the form item to select the menu option **Convert Widget**. Select a new widget from the list.

Convert a container

Right-click on the container object in the Form Design window to select the menu option **Convert Container**. Select a new container type and options in the **Convert Container** dialog.

Convert to matrix

Right-click on a field to convert it to a matrix container.

Command-line syntax: gsform

The `gsform` tool is used at the command line to compile and import forms, convert `4fd` files, extract localized strings, and keep a copy of the `per` file created during compilation.

The Genero Studio form definition file (`4fd`) can also be compiled from the operating system command line, using the tool `gsform`, which is located in the `<Studio-install-dir>/bin` directory. The `gsform` tool executable file should not be moved to another directory.

Use `gsform` if you want to:

- compile a `4fd` file from the command line
- import a `per` file into Genero Studio Form Designer format
- convert `4fd` files in the old Genero Studio format to the new `4fd` format
- extract localized strings
- keep a copy of the `per` file created during compilation

gsform Syntax

```
gsform [options] file1[4fd] file2[4fd] ...
```

where *filen* is a Genero Studio form file with or without the extension being specified.

Table 136: gsform options

Option	Description
-h	Display help instead of the standard behavior.
-m	Extract localized messages, See the <i>Genero Business Development Language User Guide</i> topic Localized Strings for additional information.
-V	Version information
-c	Convert old format <code>4fd</code> files to new format. Old format <code><filename>4fd</code> files are backed up in <code><filename>.bak</code> , New format files are saved in <code><filename>4fd</code> . If the backup fails, the conversion is aborted.
-import	Import <code>per</code> file as Genero Studio <code>4fd</code> file
-keep	Keep intermediate <code>per</code> file that is created; do not delete. The compilation process uses temporary <code>per</code> files. In case there are unexpected (internal) errors, this option avoids the temporary files deletion so that the file can be read. Important: Form Designer and <code>gsform</code> don't allow compilation when <code>per</code> files

Option	Description
	with the same file name as 4fd files are present in the same directory.
-dbname <i>database</i>	Provide the database name <i>database</i> to use for form compilation. If the 4fd files use a database, it is strongly recommended that you provide the database name using this option to reduce computation time.
-i	Ignore existing <code>per</code> file; overwrites it.
-M	Display all compilation messages. See the <i>Genero Business Development Language User Guide</i> topic Compiling Programs for additional information.
-W	Display warning messages. Option <i>all</i> turns on all messages.
-ag-templatePath	Specifies which Application Generator template directory used to load the Application Generator settings (file types, properties).

Localizing your form

Localized Strings allow you to customize the text displayed by your application, for internationalization or site-specific text.

The **localizedStr** property allows you to set localization for an individual widget or form item, where applicable. Setting the checkbox for localized strings to "True" will mark the related strings for localization.

Once the form definition is completed, you can use the [gsform](#) utility from the command line to generate a text file of the localized strings. For example:

```
gsform -m myform.4fd > myfile.str
```

The format of the file will be "string key" = "string text". You can copy the file and change the string text as needed for localization. Do not change the string key, as this is generated by Genero Studio.

The localized strings file must be added to a library node in your project in Project Manager, and the library linked to your application node. When the application is built, the localized string will be compiled and linked into your application.

See the topic on Localized Strings in the *Genero Business Development Language User Guide*.

Form Designer Reference

Reference information for Form Designer.

- [Customize Form Designer: preferences](#) on page 444
- [Form tab](#) on page 445
- [Business Record diagram](#) on page 412
- [Menus](#) on page 446
- [Views](#) on page 447
- [Dialogs](#) on page 449
- [Properties list](#) on page 459
- [Form Designer error messages](#) on page 484
- [Business Records error messages](#) on page 280
- [XML validation error messages](#) on page 497

Customize Form Designer: preferences

Information about Form Designer preferences.

New Form section

Table 137: New Form preferences

Property	Description
Default width	Default width, in number of characters.
Default height	Default height, in number of lines.

Import section

The items in this section refer to forms that are imported from the .per format.

Table 138: Import preferences

Open form after import checkbox	If checked, form will be opened in Form Designer after being imported.
Convert Text into Label checkbox	<p>If checked, forms imported will have text converted into a static label.</p> <p>Tip: It is preferred to use static labels instead of text to provide field and widget labels on a form. When you use static labels, you gain the use of all properties supported by the static label item type, to include the use of localized strings and internationalization. It is recommended that you leave this option selected.</p>

- [Form elements settings](#) on page 444
- [Database form settings](#) on page 444

Form elements settings

Table and Tree Settings

- **Default rows** - number of rows
- **Default columns** - number of columns
- **Default column width** - size for a new column, in characters

Database form settings

Container

Choose the container for the form.

Table and Tree

- **Row count** - number of rows
- **Maximum column width** - in characters

Grid and ScrollGrid

Label and Field Alignment: Icons allow you to specify:

- Label left and field left

- Label right and field right
- Label left and field right
- Label right and field left

Property label and Field:

- **Maximum width** - in characters
- **Number of fields** - number of fields per line
- **Top border** -; in lines
- **Bottom border** - in lines
- **Left margin** - in characters
- **Right margin**
- in characters
- **Field gutter** - spaces between label and data for a field
- **Field gap** - gap between two fields on the same line

TextEdit:

- **Maximum widget height** - enter value

Form tab

The Form tab displays the form in design mode.

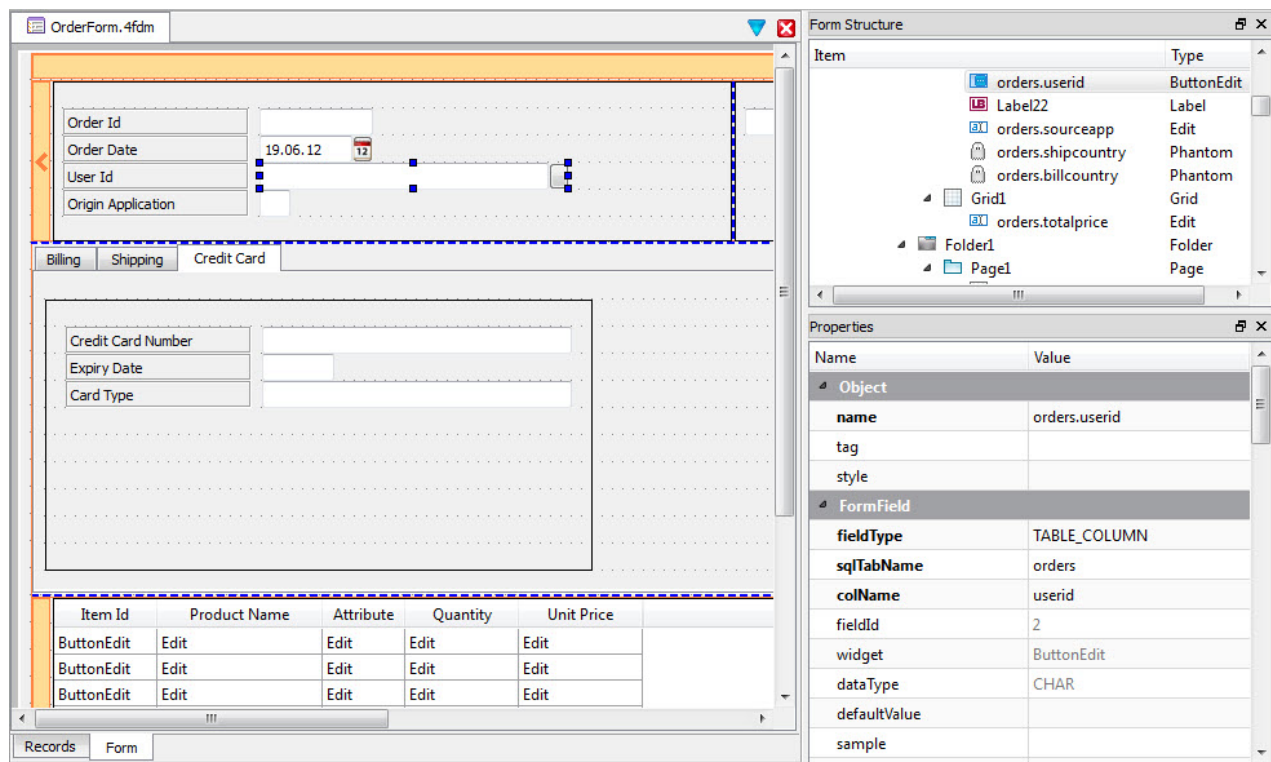


Figure 263: Form tab

Business Record diagram

The Business Record diagram is used to define the data set of the form, report, or web service.

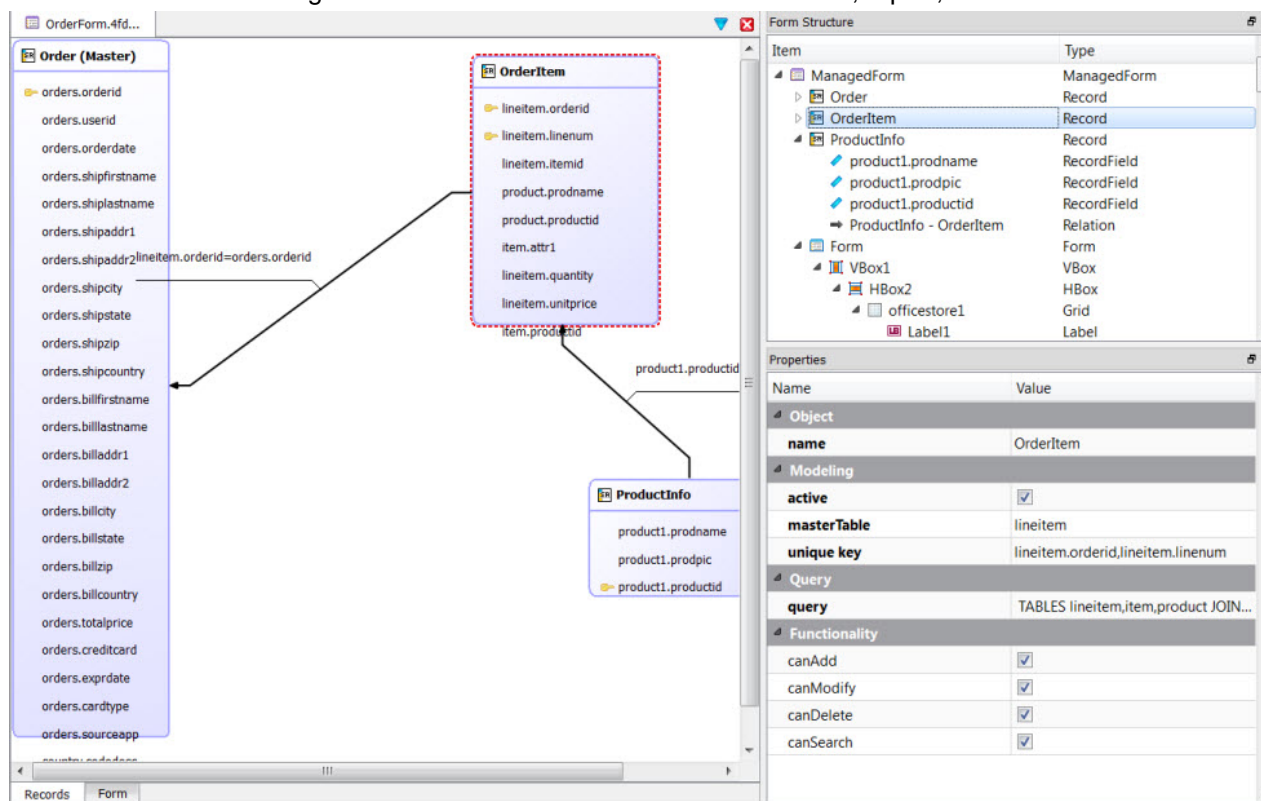


Figure 264: Business Record diagram

Menus

Information about Form Designer menus.

- [Form Designer context menu](#) on page 446
- [Alignment menu](#) on page 447

Form Designer context menu

In Form Designer, each container and widget on a form has a context menu accessible by right-clicking on the object.

The context menu displays actions applicable to the object selected.

Table 139: Right-click Context Menu

Option	Description
Edit Text	Edit the text property.
Add Phantom	Add a phantom field.
Convert Container	Convert container to a different container type.
H-Layout	Nest container in Horizontal layout .
V-Layout	Nest container in Vertical layout .
Break Layout	Remove Horizontal or Vertical layout.
Select All Form Fields	Select all form fields in the current container.

Option	Description
Add Page	Add Page to Folder container.
Add Page Before	Add Page before selected Page.
Add Page After	Add Page after selected Page.
Add Record	Add field to existing record or add field to a new record.
Add Column	Add column to table .
Add Column Before	Add column to Table before selected column.
Add Column After	Add column to Table after selected column.
Convert Widget	Convert widget to a different widget type.
Convert to Text	Convert widget to a Text widget.
Convert to Matrix	Convert a formField to matrix .
Edit Items	Edit the items property.
Locate in design view	From form structure node, locate and select item in design.

Alignment menu

The Alignment menu provides options for aligning widgets on a form.

Table 140: Alignment Menu

Option	Description
Left	Aligned with the leftmost widget selected.
Right	Aligned with the rightmost widget selected.
Center	Centered vertically based on the average centering of all selected items.
Top	Aligned with the topmost widget selected.
Bottom	Aligned with the bottommost widget selected.
Middle	Centered horizontally to the average centering of all selected items.
Advanced horizontal	See Alignment dialog on page 457
Advanced vertical	See Alignment dialog on page 457
Distribute Horizontally	Aligned with the topmost widget selected (= top alignment) AND items are evenly distributed.
Distribute Vertically	Aligned with the leftmost widget selected (= left alignment) AND items are evenly distributed

Views

Information about Form Designer views.

- [Properties view](#) on page 448
- [Structure view](#) on page 448

Properties view

Selecting one item from the design window makes this item current, and its properties are displayed in the **Properties view** and available for updating. Each form item, as well as the form itself, has properties that can be set.

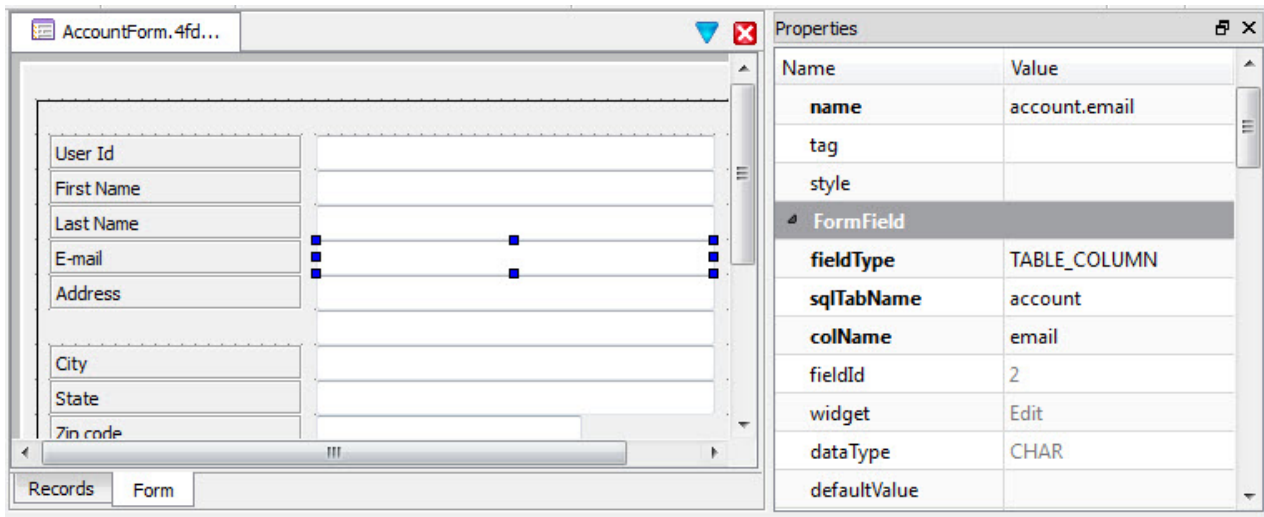


Figure 265: Properties view

Multiple properties can be selected using Ctrl-click. If multiple properties have been selected, only the common properties will display in the Properties view.

Structure view

The structure of the form is displayed in the Form Structure view. As you add form items to the Form Design window, the elements are automatically added to a tree structure in the Form Structure view.

Expand the nodes in the tree to display or hide the different form items.

Selecting a form item node also selects the corresponding object in the Form Design window, and displays the item's properties in the Properties view.

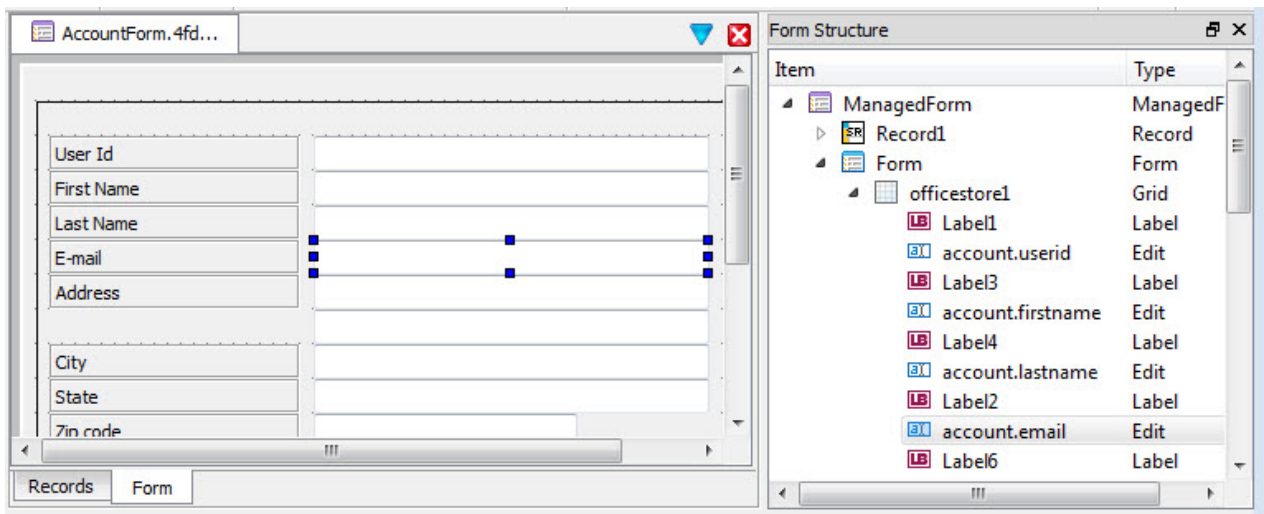


Figure 266: Structure View

Managed Form

Lists a node for each record and for the form.

Record(s)

Lists a node for each field in the record.

Forms

Lists a node for each item on the form such as containers and widgets.

Right-click on the Form node in the Structure view to add or import other form elements like a Toolbar, Topmenu, Action Defaults and Styles.

Dialogs

Information about Form Designer dialogs.

- [Data Control wizard](#) on page 449
- [Style Selection dialog](#) on page 455
- [Convert Container dialog](#) on page 456
- [Alignment dialog](#) on page 457
- [Dynamic properties](#) on page 454

Data Control wizard

The Data Control wizard is used to rapidly create a form from a database meta-schema.

Once you have [added a database schema](#) to your project, you can add fields for database columns to your form.

Access the Data Control wizard from the Form Designer or Business Application diagram:

- Form Designer: **Container >> Data Control**
- **File >> New >> Design, CRUD Form from Database** or **Genero Files, Form from Database**
- BA diagram: Right-click on a **Form** entity and select **Implement Form from Database**
- BA diagram: Right-click on a **Zoom** entity and select **Implement Zoom from Database**.

Column selection

Use the **Column selection** dialog to choose a database and select fields for the form.

To use Column Selection

1. Select the database schema from the list of schemas that you have [added to your project](#).
2. Select a table or view name in the schema list to see its columns in the **Column description** list. If a schema is not available in the list, [add it to the project first](#).
3. [Expand and/or filter the Tables list](#).
4. Select the desired columns and use the right arrows to transfer the columns to the **Selected Fields** list.
5. Use the up/down arrows to rearrange the column order. Use the left arrows to remove columns from the **Selected Fields** list.
6. Click **Next** to continue to the [Container Selection](#) page.

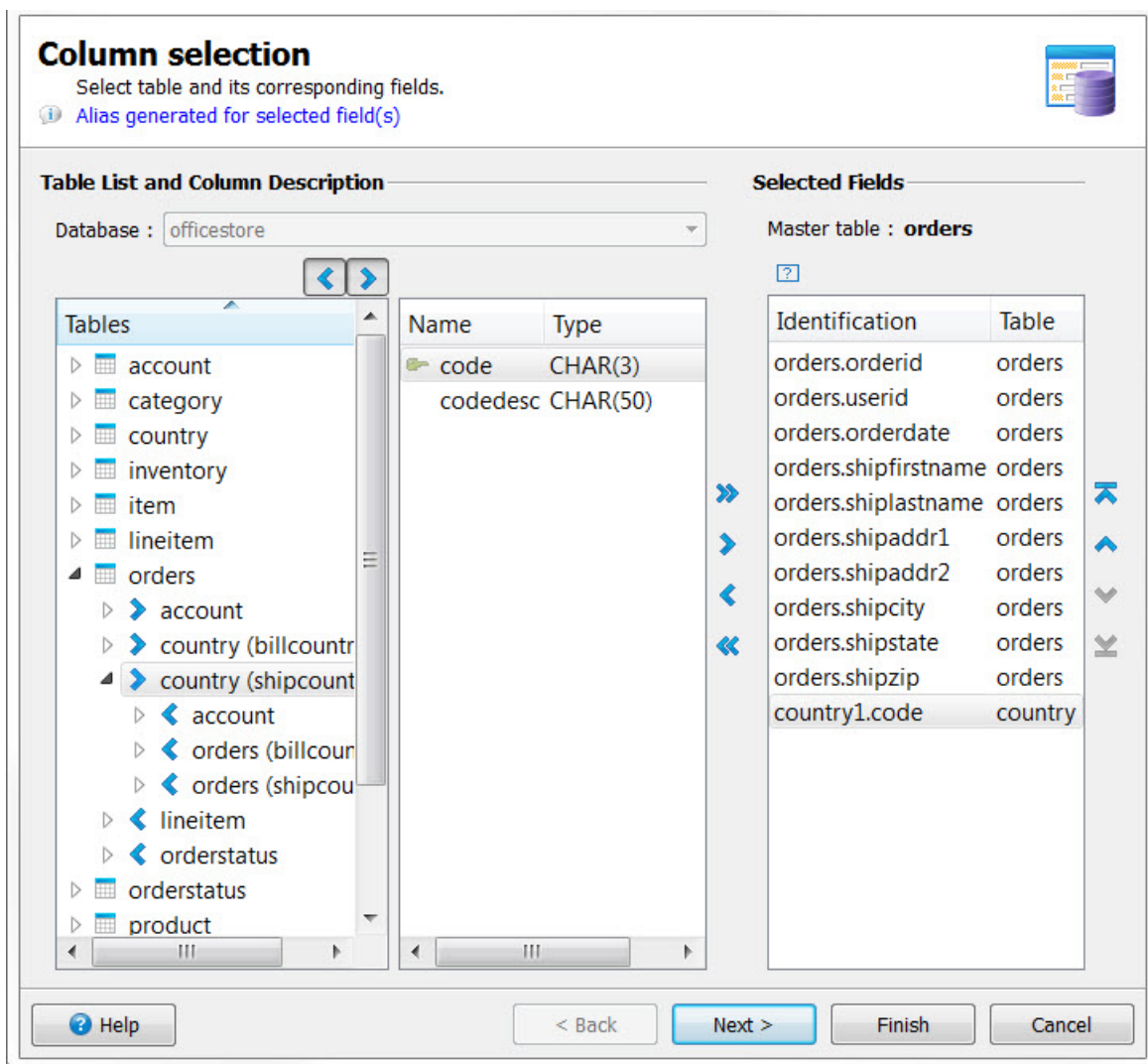


Figure 267: Data Control Wizard

Database

Available schemas.

Tables

Tables are listed in alphabetical order. Tables that have defined relationships in the schema can be expanded to show the tables to which they relate and how they relate. The **Incoming** arrow and **Outgoing** arrow buttons at the top of the **Tables** list filter the list to show the outgoing and incoming relationships:

- Incoming - Identifies those tables that have a relationship with the selected table where the selected table includes the foreign key(s).
- Outgoing - The default filter. Identifies those tables include a foreign key relationship to the selected master table.

The joins are automatically built based on the schema relationships and can be viewed or

modified in the [Joins and Data order](#) on page 453 page of the wizard.

Master table

Identifies the master table for the selected fields.

Container selection

The **Container selection** dialog lets you choose the type of container for the form.

Container selection
Select container and layout fields.

Container

Grid ScrollGrid Table Tree

Grid and ScrollGrid

Label and Field Alignment Label and Field Properties

label left and field left

Maximum width : 30 Number of fields : 2
Top border : 1 Bottom border : 1
Left margin : 1 Right margin : 1
Field gutter : 1 Field gap : 1

Matrix

Repeat

Row count : 3
Column count : 1

? Help < Back Next > Finish Cancel

Container

Select the container to hold the fields for the database columns:

- [Grid - positioning](#) on page 415
- [ScrollGrid - positioning](#) on page 415
- [Table - organizing](#) on page 417
- [Tree - hierarchy](#) on page 418

Grid and ScrollGrid

Label and Field Alignment:

- Label left and field left
- Label right and field right
- Label left and field right

- Label right and field left

Label and Field Properties:

- **Maximum width** - in characters
- **Number of fields** - number of fields per line
- **Top border** - in lines
- **Bottom border** - in lines
- **Left margin** - in characters
- **Right margin** - in characters
- **Field gutter** - spaces between label and data for a field
- **Field gap** - gap between two fields on the same line

Matrix

- **Repeat** - check to repeat the fields
- **Row count** - number of rows in matrix
- **Column count** - number of columns in matrix

Joins and Data order

The **Edit query** dialog allows you to specify the joins between the tables in a record used for a form, report, or service. Joins between tables referenced in the form are set up in the `query` property of the business record.

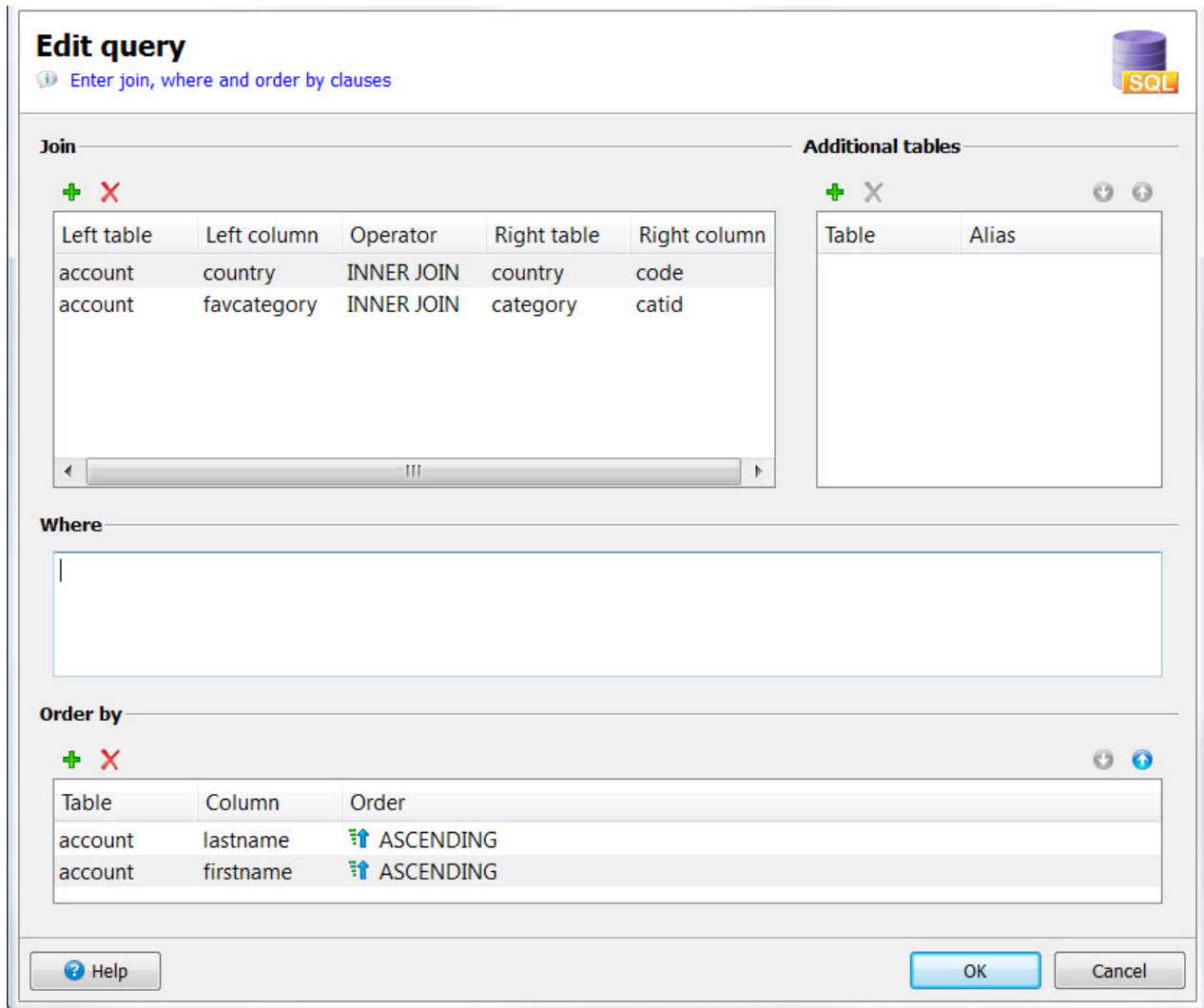


Figure 268: Edit query dialog

Join

If two tables will be involved in the query, this section is enabled to enter an SQL Join condition for the SQL Query.

Additional tables

Add a table here that does not have any fields in the record. For example, list the orders table here if the orders table is required to enable the join between the customer and items tables.

Where

Add additional conditions to the SQL Where clause, to restrict the rows returned by the SQL Query to a subset of the data.

Order by

Change the default order in which the SQL Query will return the rows.

Dynamic properties

The **Dynamic properties** dialog lets you change the label displayed for a formField or the widget used to display the value.

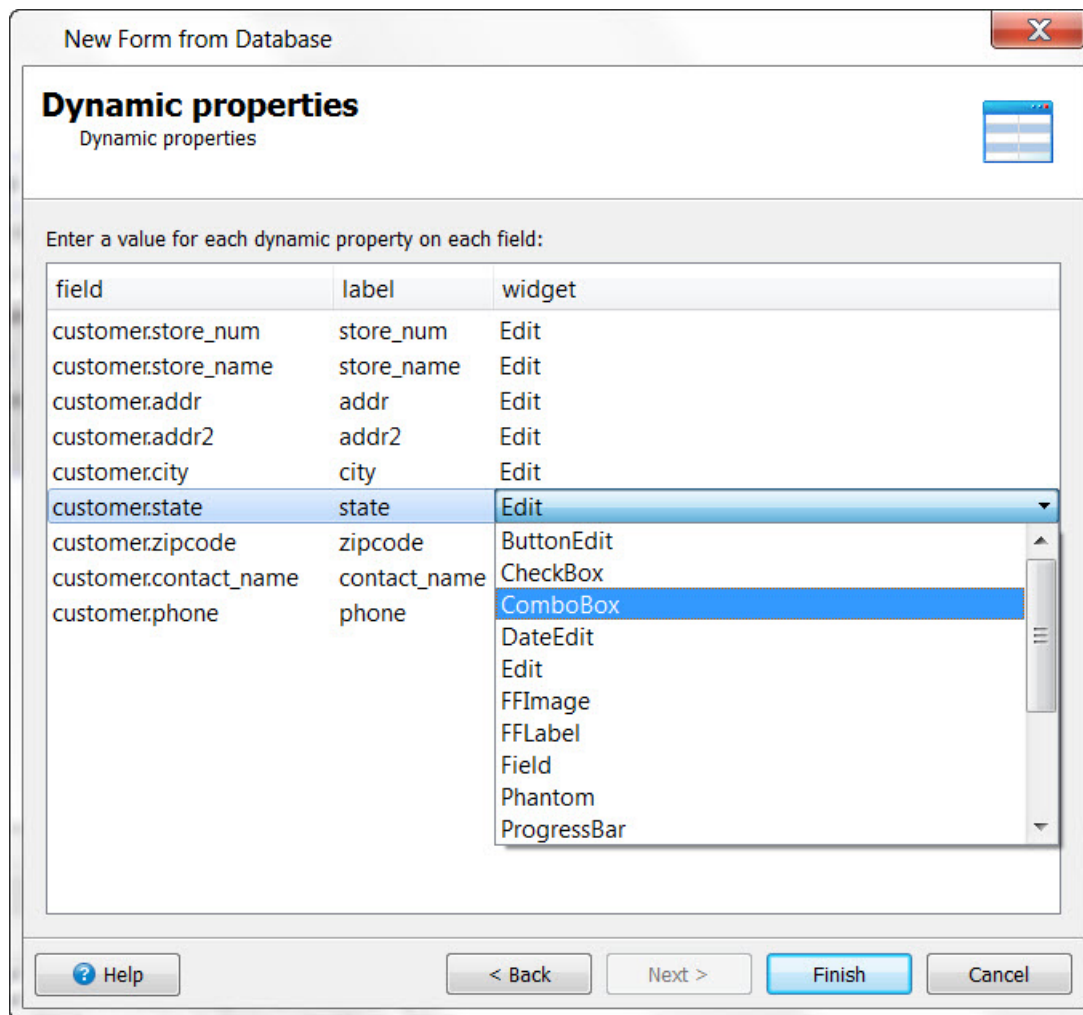


Figure 269: Dynamic properties Dialog

Select a field's label or widget property to change it.

Click **Finish** and the database columns and container are added to the form.

Style Selection dialog

The **Style Selection** dialog displays available styles from the form's style file.

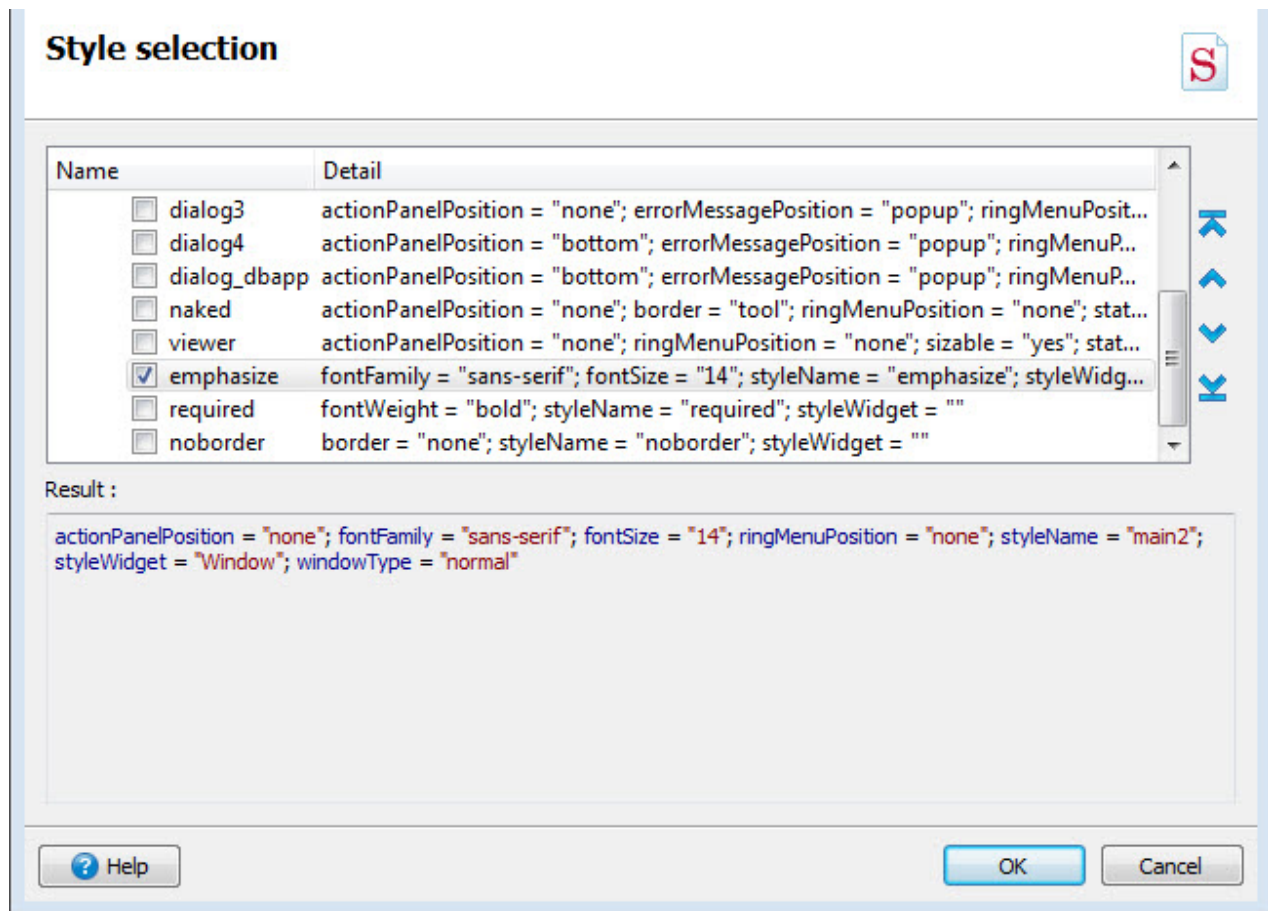


Figure 270: Style dialog

Name	Name of the style in the styles file (4st).
Detail	The definition of the style.
Result	The attributes resulting from the selection.

Check the styles you want to use. Check or uncheck the file name to select/de-select all styles.

Use the Up/Down arrows to change the priority order of the styles if the same property is defined in several styles selected.

See the Presentation Styles topic in the *Genero Business Development Language User Guide*.

Convert Container dialog

The **Convert Container** dialog displays available containers to which the selected container can be converted.

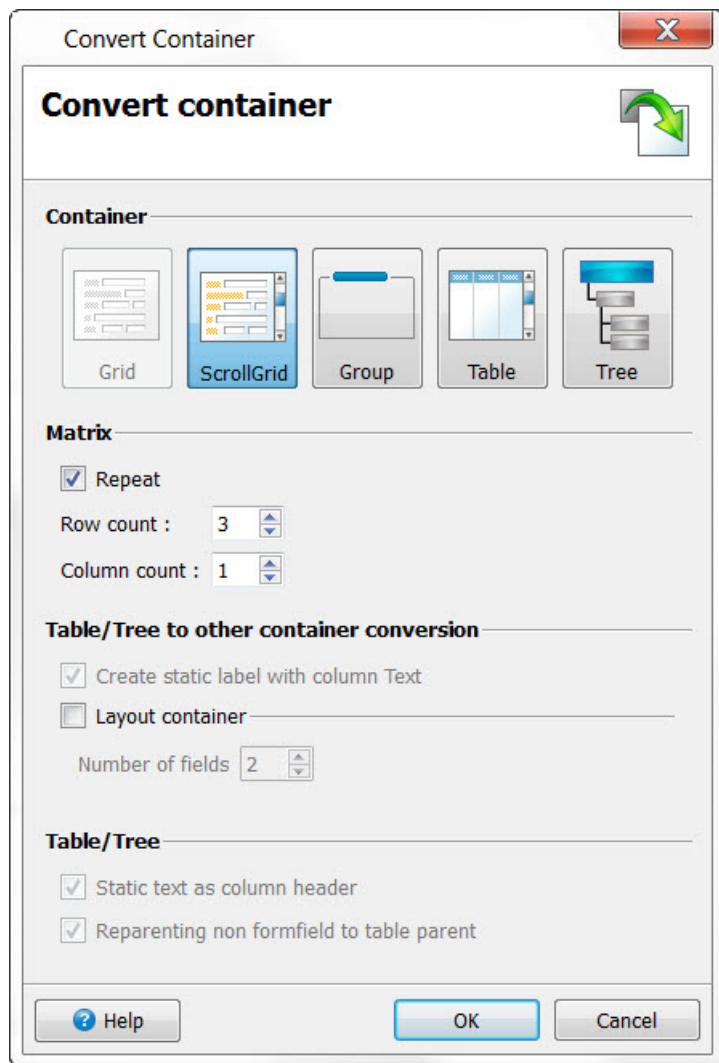


Figure 271: Convert Container dialog

Container

Select the container to hold the fields for the database columns:

- [Grid - positioning](#) on page 415
- [ScrollGrid - positioning](#) on page 415
- [Group - grouping](#) on page 416
- [Table - organizing](#) on page 417
- [Tree - hierarchy](#) on page 418

Matrix

- Repeat - check to repeat the fields
- Row count - number of rows in matrix
- Column count - number of columns in matrix

Table/Tree to other conversion

- Number of fields - indicates the number of formfield columns on each line

Alignment dialog

The Alignment dialogs provide advanced settings for item alignment in the form.

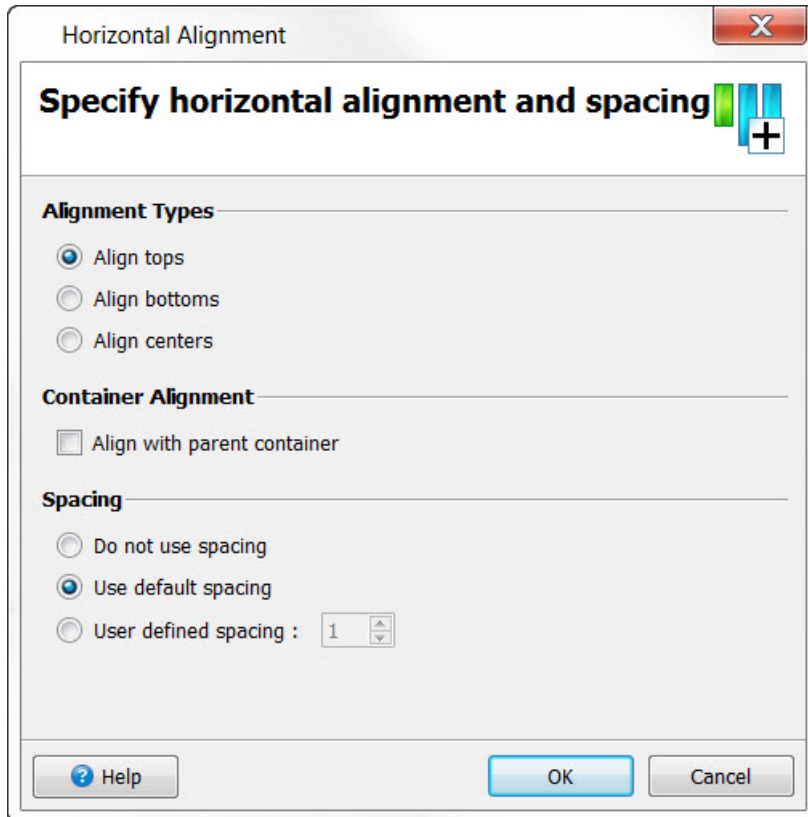


Figure 272: Horizontal alignment dialog

Alignment Types

Choose whether to align tops, bottoms, or centers of selected items.

Container Alignment

Check to align selected item(s) with parent container's border.

Spacing

Choose whether to not use spacing, use default spacing, or define spacing by the number of columns (horizontally) to put between selected items.

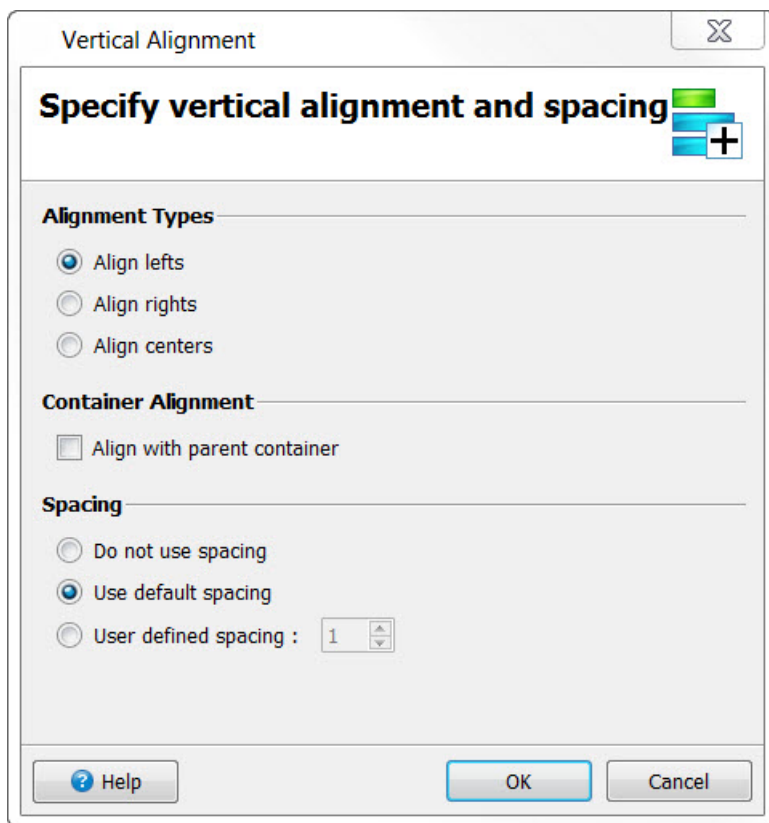


Figure 273: Vertical alignment dialog

Alignment Types

Choose whether to align lefts, rights, or centers of selected items.

Container Alignment

Check to align selected item(s) with parent container's border.

Spacing

Choose whether to not use spacing, use default spacing, or define spacing by the number of lines (vertically) to put between selected items.

Dynamic properties

The **Dynamic properties** dialog lets you change the label displayed for a formField or the widget used to display the value.

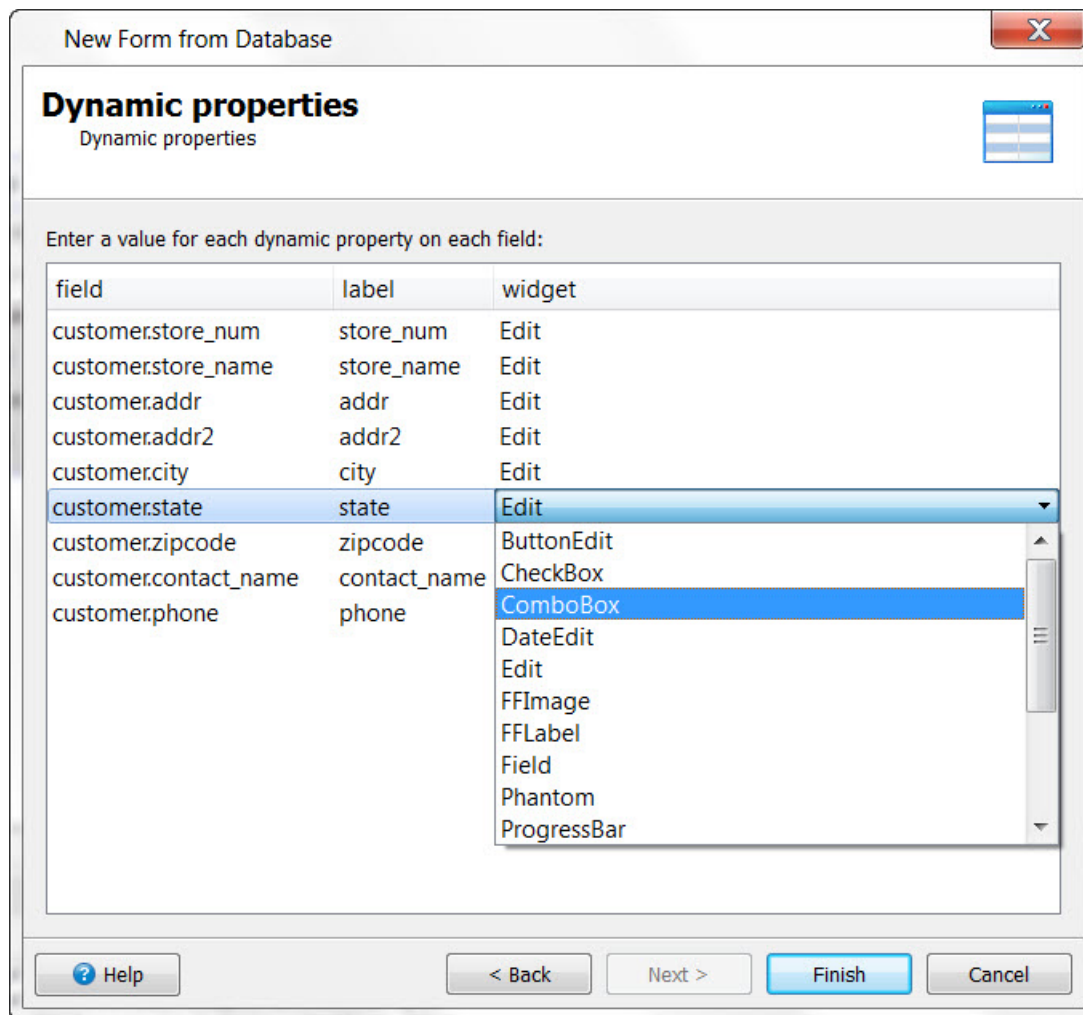


Figure 274: Dynamic properties Dialog

Select a field's label or widget property to change it.

Click **Finish** and the database columns and container are added to the form.

Properties list

Information about Form Designer properties.

- [accelerator](#) on page 462
- [accelerator2](#) on page 462
- [accelerator3](#) on page 462
- [accelerator4](#) on page 462
- [action](#) on page 462
- [aggregateText](#) on page 462
- [aggregateType](#) on page 462
- [autoNext](#) on page 462
- [autoScale](#) on page 463
- [buttonTextHidden](#) on page 463
- [case](#) on page 463

- [century](#) on page 463
- [color](#) on page 464
- [colName](#) and [sqlTabName](#) on page 464
- [colorCondition](#) on page 464
- [comment](#) on page 464
- [completer](#) on page 464
- [componentType](#) on page 465
- [contextMenu](#) on page 465
- [databaseName](#) on page 465
- [dataType](#) on page 465
- [defaultValue](#) on page 465
- [defaultView](#) on page 466
- [Display Like](#) on page 466
- [doubleClick](#) on page 466
- [expandedColumn](#) on page 467
- [fieldType](#) on page 467
- [fontPitch](#) on page 467
- [format](#) on page 467
- [gridChildrenInParent](#) on page 470
- [gridHeight](#), [gridWidth](#) on page 470
- [hidden](#) on page 470
- [idColumn](#) on page 471
- [image](#) on page 471
- [imageCollapsed](#) on page 472
- [imageColumn](#) on page 472
- [imageExpanded](#) on page 473
- [imageLeaf](#) on page 473
- [include](#) on page 473
- [initializer](#) on page 473
- [invisible](#) on page 473
- [isNodeColumn](#) on page 474
- [items](#) on page 474
- [justify](#) on page 474
- [keyboardHint](#) on page 475
- [minHeight](#), [minWidth](#) on page 475
- [name](#) on page 475
- [noEntry](#) on page 475
- [notNull](#) on page 476
- [orientation](#) on page 476
- [parentIdColumn](#) on page 476
- [PER comments](#) on page 476
- [picture](#) on page 476
- [posX](#), [posY](#) on page 477
- [program](#) on page 477
- [queryEditable](#) on page 477
- [required](#) on page 477
- [reverse](#) on page 478
- [rowHeight](#) on page 478
- [sample](#) on page 478
- [scroll](#) on page 478

- [scrollbars](#) on page 478
- [sizePolicy](#) on page 478
- [sliderOrientation](#) on page 479
- [spacing](#) on page 479
- [splitter](#) on page 479
- [step](#) on page 480
- [stretch](#) on page 480
- [style](#) on page 480
- [styleFile](#) on page 480
- [tabIndex](#) on page 480
- [tag](#) on page 480
- [text](#) on page 481
- [title](#) on page 481
- [totalRows](#) on page 481
- [unHidable](#) on page 481
- [unHidableColumns](#) on page 481
- [unMovable](#) on page 481
- [unMovableColumns](#) on page 481
- [unSizable](#) on page 481
- [unSizableColumns](#) on page 482
- [unSortable](#) on page 482
- [unSortableColumns](#) on page 482
- [validate](#) on page 482
- [Validate Like](#) on page 482
- [valueChecked](#) on page 482
- [valueMax](#) on page 483
- [valueMin](#) on page 483
- [valueUnchecked](#) on page 483
- [verify](#) on page 483
- [version](#) on page 483
- [wantFixedPageSize](#) on page 484
- [wantNoReturns](#) on page 484
- [wantTabs](#) on page 484
- [windowStyle](#) on page 484

accelerator

The `accelerator` property defines the `_primary` accelerator key of an action default item.

accelerator2

The `accelerator2` property defines the secondary accelerator key of an action default item.

accelerator3

The `accelerator3` property defines the third accelerator key of an action default item.

accelerator4

The `accelerator4` property defines the fourth accelerator key of an action default item.

action

The `action` property defines the name of the action to be sent to the program when the user activates the form item.

aggregateText

The `aggregateText` property can be used to define the label to be displayed for aggregate fields.

Usage

The `aggregateText` property can be specified at the aggregate field level, or globally at the [Table - organizing](#) on page 417 level, to define a label for the whole summary line. When defining the `aggregateText` property at the aggregate field level, the text will be anchored to the value cell. If the `aggregateText` property is specified at the Table level, the label will appear on the left in the summary line. When an aggregate text is defined at both levels, the global aggregate text of the table will be ignored.

aggregateType

The `aggregateType` property defines how the aggregate field value is computed.

Usage

PROGRAM specifies that the aggregate value will be computed and displayed by the program code.

An aggregate type different from PROGRAM specifies that the aggregate value is computed automatically:

- SUM computes the total of all values of the corresponding numeric column.
- AVG computes the average of all values of the corresponding numeric column.
- MIN displays the minimum value of the corresponding numeric column.
- MAX displays the maximum value of the corresponding numeric column.
- COUNT computes the number of rows.

The SUM and AVG aggregate types apply to data types that can be used as operand for an addition, such as INTEGER, DECIMAL, INTERVAL.

The MIN and MAX aggregate types apply to data types that can be compared, such as INTEGER, DECIMAL, INTERVAL, CHAR, DATETIME.

autoNext

The `autoNext` property causes the cursor to automatically advance during input to the next field when the current field is full.

Usage

If data values entered in the field do not meet the requirements of other field properties like INCLUDE or PICTURE, the cursor does *not* automatically move to the next field but remains in the current field, and an error message displays.

The `autoNext` property is particularly useful with character fields in which the input data is of a standard length, such as numeric postal codes or the abbreviations in the **state** table. It is also useful if a character field has a length of 1 because only one keystroke is required to enter data and move to the next field.

autoScale

The `autoScale` property causes the form element contents to automatically scale to the size given to the item.

Usage

For an [Image](#) on page 426 this property forces the image to be stretched to fit in the area reserved for the image.

buttonTextHidden

The `buttonTextHidden` property indicates that the labels of the buttons of the element should not be displayed.

Usage

Use in a [Toolbar](#) definition to hide the labels of buttons.

case

The `case` property forces character input to uppercase or lowercase letters.

Usage

Assign the `case` property to a character field when you want the runtime system to convert to uppercase or lowercase letters entered, both on the screen and in the corresponding program variable.

Because uppercase and lowercase letters have different values, storing character strings in one or the other format can simplify sorting and querying a database.

Characters entered by the user are converted in INPUT, INPUT ARRAY, and CONSTRUCT instructions.

The results of conversions between uppercase and lowercase letters are based on the locale settings (LANG). When using single byte runners, the conversion of ASCII characters >127 is controlled by the LC_CTYPE environment variable.

century

The `century` property specifies how to expand abbreviated one- and two-digit year specifications in a DATE and DATETIME field.

Purpose

Usage

Expansion is based on this setting (and on the year value from the system clock at runtime).

The `century` property can specify any of four algorithms to expand abbreviated years into four-digit year values that end with the same digits (or digit) that the user has entered.

`century` supports the same settings as the DBCENTURY environment variable, but with a scope that is restricted to a single field.

If the `century` and DBCENTURY settings are different, CENTURY takes precedence.

Unlike DBCENTURY, the `century` property is not case sensitive. However, we recommend that you use uppercase letters in the property.

color

The `color` property defines the foreground color of the text displayed by a form element.

Usage

The `color` property defines the logical color of a value displayed in a field. Value can be BLACK, BLUE, CYAN, GREEN, MAGENTA, RED, WHITE, and YELLOW.

For backward compatibility, the value can be combined with an intensity keyword: REVERSE, LEFT, BLINK, and UNDERLINE.

colName and sqlTabName

The `colName` property is the name of the database column, the `sqlTabName` property is the name of the database table for formField form items.

colorCondition

The `colorCondition` property defines a condition to set the foreground color dynamically, though it is recommended that you use styles to implement conditional colors.

Usage

The `colorCondition` property defines the logical color of the text of a field when the value satisfies the conditional expression.

The Expression Editor allows you to create the expression for which the `colorCondition` is evaluated.

The condition in `colorCondition` can only reference the field for which the property is set. The Boolean expression is automatically evaluated at runtime to check when the color property must be set.

Example Expressions

```
COLOR = GREEN WHERE today
```

To refer to the value in the expression, use the keyword `$VALUE`:

```
COLOR = RED WHERE $VALUE > 100
```

comment

The `comment` property defines text that can be shown in the comment line when the form item becomes current.

Usage

The most common use of the `comment` property is to give information or instructions to the user. This is particularly appropriate when the field accepts only a limited set of values.

The screen location where the message is displayed depends on external configuration. It can be displayed in the COMMENT LINE, or in the STATUSBAR when using a graphical user interface.

If the OPEN WINDOW statement specifies `COMMENT LINE OFF`, any output to the comment area is hidden even if the window displays a form that includes fields that include the `COMMENT` property.

See the topic on Statusbar in the *Genero Business Development Language User Guide*.

completer

The `completer` property enables autocompletion for the edit field.

Form fields with `COMPLETER` attribute provide suggestions while the end-user types text into the field, it can be used in text edit fields such as EDIT and BUTTONEDIT item types.

For more information, see the *COMPLETER attribute* topic in the *Genero Business Development Language User Guide*.

componentType

The `componentType` property defines a name identifying the external widget.

Usage

The `componentType` property is used to define the type of a [WebComponent](#) form item.

The value of this property will be mapped to a specific widget definition on the front-end side. See front-end specific documentation related to Web Components.

contextMenu

The `contextMenu` property defines the action default property whether a context menu option must be displayed for an action.

Usage

contextMenu values:

1. `NO` indicates that no context menu option must be displayed for this action.
2. `YES` indicates that a context menu option must always be displayed for this action, if the action is visible (`setActionHidden` method).
3. `AUTO` means that the context menu option is displayed if no explicit action view is used for that action and the action is visible (`setActionHidden` method).

The default is `YES`.

This property applies to the actions defined by the current dialog in the current window.

databaseName

The `databaseName` property specifies the name of the database shown in the DB Schemas tab.

dataType

The `dataType` property specifies the data type of the item.

Usage

For `NON_DATABASE` `formField` items, the data type has to be specified. For other `formField` items, data type is determined from the database column type and cannot be modified.

defaultValue

The `defaultValue` property assigns a default value to a field during data entry.

Usage

The effect of the `defaultValue` property depends on the `WITHOUT_DEFAULTS` configuration option of the dialog using the form:

- With the `INPUT` statement, form default values have are ignored when using the `WITHOUT_DEFAULTS` option. With this option, the runtime system displays the values in the program variables to the screen. Otherwise, the form default values will be displayed when the dialog starts.
- With the `INPUT ARRAY` statement, the form default values are always used for new rows inserted by the user. With `INPUT ARRAY`, the `WITHOUT_DEFAULTS` option indicates if the existing program array elements have to be used.

Defaults values can also be specified in the database schema file, for `formFields` defined with database column reference.

If the field is formonly (NON_DATABASE), you must also specify a data type when you assign the default property to a field.

If both the default property and the [required](#) property are assigned to the same field, the required property is ignored.

DATETIME and INTERVAL literals are not supported.

defaultView

The `defaultView` property defines the action default property whether a default view (i.e. button) must be displayed for a given action.

Usage

defaultView values:

- NO indicates that no default action view must be displayed for this action.
- YES indicates that a default action view must always be displayed for this action, if the action is visible (`setActionHidden`).
- AUTO means that a default action view is displayed if no explicit action view is used for that action and the action is visible (`setActionHidden`).

The default is AUTO.

This property applies to the actions defined by the current dialog in the current window.

Display Like

The `Display Like` property takes column properties defined in the database schema files and applies them to a field.

Usage

Specifying this property is equivalent to listing all the properties that are assigned to *table.column* in the database schema file generated from the **syscolatt** table.

Supply the `displayTabName` for the table name and `displayColName` for the column name.

Display properties are automatically taken from the schema file if the field is linked to a *table.column*.

The `Display Like` value is evaluated at compile time, not at runtime. If the database schema file changes, you might need to recompile a program that uses the `LIKE` clause. Even if all of the fields in the form are `FORMONLY`, this property requires the form compiler to access the database schema file that contains the description of *table*.

doubleClick

The `doubleClick` property defines the name of the action to be sent when the user double-clicks on a Table row.

Usage

This property is typically used in a [table](#) container, to define the action to be sent when the user double-clicks on a row. By default, if the Table is driven by a `DISPLAY ARRAY`, a double-click invokes the *accept* action. When using an `INPUT ARRAY`, double-click selects the whole text if the current widget is editable. If `doubleclick` is defined when using an `INPUT ARRAY`, the action can only be sent when the user double-clicks on a non-editable widget like a [label](#).

expandedColumn

The `expandedColumn` property specifies the `formField` that indicates whether a tree node is expanded (opened). This property is optional.

Usage

This property is used in the definition of a [container](#), see [Tree Views](#) for more details.

fieldType

The `fieldType` property specifies the category of the values stored in the item.

Purpose

`fieldType` values:

- **TABLE_COLUMN**: the value is defined in terms of a database column. Provide the **sqlTabName**, the name of the database table. Provide the **colName**, the name of the database column.
- **NON_DATABASE**: the value is not defined in terms of a database column .
- **COLUMN_LIKE** : the value is like **NON_DATABASE**, but takes its description from a database column.
- **TABLE_ALIAS**: the value is defined in terms of an alias that is assigned to a database table.

fontPitch

The `fontPitch` property defines the character font type as fixed or variable when the default font is used, though it is recommended that you use styles to define font types.

Usage

By default, most front ends use variable width character fonts, but in some cases you might need to use a fixed font.

When using `FIXED`, you force the characters to have a fixed size.

When using `VARIABLE`, you allow the characters to have a variable size.

format

The `format` property controls the format of numeric and date time fields for output displays.

Usage

Supply the `format-string` for the `format` property.

Note:

1. *format-string* is a string of characters that specifies a data display format.
2. You must enclose *format-string* within quotation marks (").
3. If *format-string* is smaller than the field width, you get a compile-time warning, but the form is usable.

The `format` property can be set to define a display format for numeric and date fields. When this property is not used, environment variable settings define the default format. For `MONEY` and numeric fields such as `DECIMAL` fields, a global format can be specified with the `DBMONEY` or `DBFORMAT` environment variables. For `DATE` fields, the global format is defined by the `DBDATE` environment variable.

Understand that the `format` property is applied when displaying program variable data to `formFields`. In order to control user input with a mask, you must use the [picture](#) property instead. The `picture` property is typically used to specify an input mask for formatted character string fields.

If *format-string* is smaller than the field width, you get a compile-time warning, but the form is usable.

Numeric formats

For DECIMAL, MONEY, SMALLFLOAT, and FLOAT data types, *format-string* consists of a set of place holders that represent digits, currency symbols, thousands and decimal separators. For example, "###.##@" defines three places to the left of the decimal point and exactly two to the right, plus a currency symbol at the end of the string.

When used with numeric values, the *format-string* must use normalized place holders described in format. The place holders will be replaced by the elements defined in the DBMONEY or DBFORMAT environment variables.

Field input cannot be supported if the format is not defined with normalized place holders.

If the numeric value is too large to fit in the number of characters defined by the format, an overflow text is displayed (****).

If the actual number displayed requires fewer characters than *format-string* specifies, numbers are right-aligned and padded on the left with blanks.

If necessary to satisfy the *format-string* specification, the number values are rounded before display.

Table 141: Format-string symbols for Numeric data types

Character	Description
*	The star placeholder fills with asterisks any position that would otherwise be blank.
&	The ampersand placeholder is used to define the position of a digit, and is replaced by a zero if that position would otherwise be blank.
#	The sharp placeholder is used to define the position of a digit, it is used to specify a maximum width for the resulting string. This wildcard character does not change any blank positions in the display: The character is replaced by a blank if no digit is to be displayed at that position.
<	Consecutive <i>less than</i> characters cause left alignment and define digit positions.
-	Displays a minus sign or a blank at that position. USING displays a minus sign when the expression is lower than zero, and otherwise a blank character. When you group several minus signs in the format string, a single minus sign floats immediately to the left of the displayed number.
+	Displays a plus or minus sign at that position. USING displays a plus sign when the expression is greater than or equal to zero, and a minus sign when the value is less than zero. When you group several plus signs in the format string, a single plus sign floats immediately to the left of the displayed number.
(Displayed as left parenthesis for negative numbers. It is used to display <i>accounting parentheses</i> instead of a minus sign for negative numbers. Consecutive left parentheses display a single left parenthesis to the left of the number being printed.
)	Displayed as right parenthesis for negative numbers. This wildcard character is used in conjunction with a open brace to display <i>accounting parentheses</i> for negative numbers.
, (comma)	The comma placeholder is used to define the position for the thousand separator defined in DBFORMAT. The thousand separator will only be displayed if there is a number on the left of it.
. (period)	The period placeholder is used to define the position for the decimal separator defined in DBMONEY or DBFORMAT. You can only have one decimal separator in a number format string.

Character	Description
\$	The dollar sign is the placeholder for the <i>front</i> currency symbol defined in DBMONEY or DBFORMAT. When you group several consecutive dollar signs, a single front currency symbol floats immediately to the left of the number being printed. The front currency symbol can be defined in DBFORMAT with more than one character.
@	The at sign is the placeholder for the <i>back</i> currency symbol defined in DBMONEY or DBFORMAT. Put several consecutive @ signs at the end of the format string to display a currency symbol defined in DBFORMAT with more than one character.

Table 142: Combinations of DBFORMAT setting and FORMAT property

FORMAT property	Numeric value	DBFORMAT	Result string
---,--&.&&	-1234.56	:.:.:	-1.234.56
\$---,--&.&&	-1234.56	E:.:.:	E -1.234,56
---,--&.&&@	-1234.56	:.:.:E	-1,234.56E

When the user enters numeric or currency values in fields, the runtime system behaves as follows:

- If a symbol is entered that was defined as a decimal separator in DBFORMAT, it is interpreted as the decimal separator.
- For MONEY fields, it disregards any *front* (leading) or *back* (trailing) currency symbol and any thousands separators that the user enters.
- For DECIMAL fields, the user must enter values without currency symbols.

Date formats

Table 143: Format-string symbols for DATE and DATETIME data type

Character	Description
dd	Day of the month as a 2-digit integer.
ddd	Three-letter English-language abbreviation of the day of the week. For example: Mon, Tue.
mm	Month as a 2-digit integer.
mmm	Three-letter English-language abbreviation of the month. For example: Jan, Feb.
yy	Year, as a 2-digits integer representing the 2 trailing digits.
yyyy	Year as a 4-digit number.

Any other character is interpreted as a literal and will be displayed as is in the field.

Table 144: Format-string examples and corresponding display formats for a DATE field

FORMAT property	Date value	DBDATE	Result string
None (DBDATE applies)	1999-09-23	DMY4/	23/09/1999
dd-mm-yyyy	1999-09-23	DMY4/	23-09-1999
dd-mm-yy	1999-09-23	DMY4/	23-09-99
(ddd.) mmm. dd, YYYY	1999-09-23	DMY4/	(Thu.) Sep. 23, 1999

Example

```
"mm/dd/yyyy"
```

gridChildrenInParent

The `gridChildrenInParent` property is used for a container to align its children to the parent container.

Usage

By default, child elements of a container are aligned locally inside the container layout cells. With this property, you can force children to be aligned according to the layout cells of the parent container of the container to which you assign this property.

This is useful, for example, when you want to align fields across Group containers inside a [Grid](#).

gridHeight, gridWidth

The `gridHeight` and `gridWidth` properties define the height and width of the form or form item.

hidden

The `hidden` property indicates that the element should not be displayed.

`hidden` property values:

1. `true` sets the underlying item property to 1.
2. `user` sets the underlying item property to 2.
3. `false` sets the underlying item property to 0.

Usage

By default, all elements are visible. You can use the `hidden` property to hide an element, such as a `formField` or a `groupbox`. The runtime system handles hidden `formFields`. If you write an `INPUT` statement using a hidden field, the field is ignored (as if it was declared as `noEntry`). Programs may change the visibility of `formFields` dynamically with the `ui.form` built-in class.

When set the `hidden` property to **true**, the underlying item property is set to 1. The value 1 indicates that the element is hidden to the user without the possibility of showing the element, for example with the context menu of `table` headers. In this hidden mode, the `unHidable` property is ignored by the front end.

When you set `hidden` to **user**, the underlying item property is set to 2. The value 2 indicates that the element is hidden by default, but the user can show/hide the element as needed. For example, the user can change a hidden column back to visible. Form elements like table columns that are hidden by the user

might be automatically reshown (`hidden=0`) by the front-end if the program dialog gives the focus to that field for input. In such case the program dialog takes precedence over the hidden property.

When you set a `hidden` property for a `formField`, the model node gets the hidden property, not the view node.

`formFields` hidden with the user option (value 2) might be shown anyway if the field is needed by a dialog for input.

idColumn

The `idColumn` property specifies the `formField` that contains the identifier of a tree node.

Usage

This property is mandatory. This property is used in the definition of a [Tree](#), see Tree Views for more details.

image

The `image` property defines the image resource to be displayed in the form item.

Usage:

This property is used to define the image resource to be displayed for form items such as a [button](#), [buttonEdit](#), or a [static image](#).

The `resource` string can be:

1. A simple file name (with or without extension), using a relative or an absolute path.
2. A path to an image on a server in the URL (Uniform Resource Locator) form.

It is recommended that you use simple image file names without the file extension, and define the `GSTIMAGEPATH` environment variable to centralize image files on the application server in a directory created specifically for images. For portability reasons, use `.png` or `.svg` image file formats only.

Supported image formats

Here is the list of image file formats supported by the different front-ends:

Table 145: List of image file formats supported by different front-ends

Suffix (case insensitive)	Front-ends supporting the file format
.BMP	GDC, GWC
.GIF	GDC, GWC
.ICO	GDC, GWC
.JPG	GDC, GWC
.PNG	GDC, GWC
.SVG	GDC, GWC
.TIFF	GDC, GWC

According to the front-end type, some image file formats or image data formats might not be supported.

Using file names or paths

If the image specification is a simple string without an URL or URI prefix, it is identified as a file path. The file is first sought in the picture directory on the client workstation. According to the front-end type, this local directory can actually be on a remote machine where the GAS middleware component is located. If the file

is not found, the front-end automatically sends an image request to the runtime system, in order to search for an image on the server where the programs are executed. The runtime system searches for server-side images by using the GSTIMAGEPATH environment variable. If GSTIMAGEPATH is not set, the image files are searched in the current working directory.

Important:

By default, if GSTIMAGEPATH is not set, the image files are searched in the current working directory. Image filenames can use absolute or relative paths and the whole application server file system can be searched (according to the permissions of the operating system user running the gstrun process). This can be a security hole because fake front-ends could ask for critical server files that are not images.

When setting GSTIMAGEPATH, the runtime system will only transfer files found in the directories listed in that environment variable. You can still use absolute or relative paths in the image file names, but the files must be located below one of the directories listed in GSTIMAGEPATH. For maximum security, put the image files in directories that contain only image files, and keep critical data or program file in separate directories.

Images displayed by program to [image](#) fields do not follow the GSTIMAGEPATH security restriction. Image field do not use the `IMAGE` property. For image fields, the field value specifies the image.

Using an image server with URL names

If the image specification starts with a URL prefix, the front-end will try to download the image from the location specified by the URL.

Currently supported URLs are:

Table 146: Supported image resource locations (URLs)

Image resource location (URL)	Description
<code>http://location-specification</code>	HTTP server
<code>https://location-specification</code>	HTTPS server (HTTP over SSL)
<code>ftp://location-specification</code>	FTP server

imageCollapsed

The `imageCollapsed` property sets the global icon to be used when a tree node is collapsed.

Usage

This property is optional. This property defines the icon to be used for nodes that are collapsed. It overwrites the program array image defined by [imageColumn](#), if both are used.

This property is used in the definition of a [Tree](#), see Tree Views for more details.

imageColumn

The `imageColumn` property defines the formField containing the image of a field.

Usage

This property is used in the definition of a [Tree](#), see the Tree View page for more details.

The images defined by the [imageCollapsed](#), [imageExpanded](#), [imageLeaf](#) properties take precedence over the images defined by the `imageColumn` cell.

imageExpanded

The `imageExpanded` property sets the global icon to be used when a tree node is expanded.

Usage

This property is optional. This property defines the icon to be used for nodes that are expanded. It overwrites the program array image defined by [imageColumn](#) on page 472, if both are used.

This property is used in the definition of a [Tree - hierarchy](#) on page 418, see Tree Views for more details.

imageLeaf

The `imageLeaf` property defines the global icon for leaf nodes of a Tree container.

Usage

This property is optional. This property defines the icon to be used for all leaf nodes of the tree. It overwrites the program array image defined by [imageColumn](#) on page 472, if both are used.

This property is used in the definition of a [Tree - hierarchy](#) on page 418, see Tree Views for more details.

include

The `include` property specifies acceptable values for a field and causes the runtime system to check the data before accepting an input value.

Usage

If the field is formonly, you must also specify a data type when you assign the `include` property to a field.

DATETIME and INTERVAL literals are not supported.

initializer

The `initializer` property allows you to specify an initialization function that will be automatically called by the runtime system to set up the form item.

Usage

The initialization function must exist in the program using the form file and must be defined with a `ui.ComboBox` parameter.

invisible

The `invisible` property prevents user-entered data from being echoed on the screen during an interactive statement.

Usage

Characters that the user enters in a field with the `invisible` property are not displayed during data entry. Depending on the front end type, the typed characters are displayed using the blank, star, underscore or dot characters.

The `invisible` property does *not* prevent display instructions like `DISPLAY` and `DISPLAY ARRAY` from explicitly displaying data in the field.

isNodeColumn

The `isNodeColumn` property specifies the `formField` that indicates whether a tree node has children.

Usage

This property is optional. Even if the program node does not contain child nodes for this tree node, this property may be used, to implement dynamic filling of tree views.

This property is used in the definition of a [Tree](#), see Tree Views for more details.

items

The `items` property defines a list of possible values that can be used by the form item.

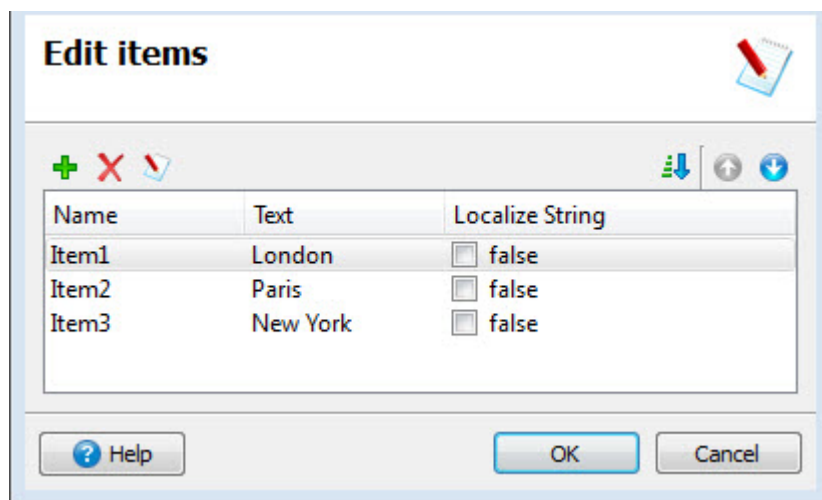
Example

Figure 275: Items Editor

Usage

This property is not used by the runtime system to validate the field, you must use the [include](#) property to force the possible values.

You can specify item labels with localized strings.

You can define a `NULL` value for an item (An empty string is equivalent to `NULL`).

justify

The `justify` property defines the justification of the content of a field and the alignment of table column headers.

Usage

With the `justify` property, you specify the justification of the content of a field as `LEFT`, `CENTER` or `RIGHT` when the field is in display state. This property is ignored for input (i.e. when the field has the focus); only the default data justification rule applies when a field is in input state. The default data justification depends on the dialog type, the field data type and the [format](#) property. For example, a numeric field value is right aligned, while a string field is left aligned. The type of dialog also defines the default justification: In a `CONSTRUCT`, all input fields are left aligned, for search criteria input.

The `justify` property can be used with all form item types. Additionally to the field content/data alignment, `justify` defines the alignment of table column headers indirectly (i.e. table column header follows the alignment of field data). However, column header alignment in tables may not be enabled by default; Check the front-end `headerAlignment` Style attribute.

keyboardHint

The `keyboardHint` property gives an indication on the kind of data the form field contains, to let the front-end adapt the keyboard accordingly.

Usage

The `keyboardHint` property can be used to give a hint to the front-end, regarding the kind of data the form field will contain. According to this hint, the front-end will open the virtual keyboard adapted to the data type, especially useful when designing application forms for mobile platforms.

Valid values for `keyboardHint` are:

- `default`: No hint, the only hint is the data type of the program variable bound to the form field.
- `email`: The field is used to enter an e-mail address.
- `number`: The field is used to enter a numeric value.
- `phone`: The field is used to enter a phone number.

For example, when defining a numeric field with the `keyboardHint` property set to `number`, the iOS device will display a numeric keyboard when entering data into that field.

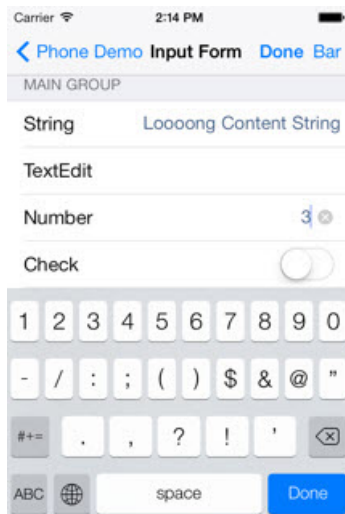


Figure 276: Mobile application using a numeric keyboard

minHeight, minWidth

The `minHeight` and `minWidth` properties define the minimum height and width of a form.

Usage

The `minHeight` and `minWidth` properties are used to define a minimum height of the form/window. This property is set on the Form.

name

The `name` property identifies the name of the item.

noEntry

The `noEntry` property prevents data entry in the field during an `INPUT` or `INPUT ARRAY` statement.

Usage

Use the `noEntry` property to bypass field input during an `INPUT` or `INPUT ARRAY` statement. When using a `WITHOUT DEFAULTS` dialog option, the content of the corresponding program variable is displayed in the field. A `noEntry` field is like a disabled field, it cannot get the focus.

The `noEntry` property does *not* prevent data entry into a field during a `CONSTRUCT` statement.

notNull

The `notNull` property specifies that the field does not accept NULL values.

Usage

This property requires that the field contains a value. If the field contains a default value, the `notNull` property is satisfied. To insist on data entry from the user, use `notNull` in the field definition, or make sure the corresponding column is defined as `notNull` in the database schema file.

The `notNull` keywords can also be used in the type definition of formonly fields.

The `notNull` property is effective only when the field name appears in the list of screen fields of an INPUT or INPUT ARRAY.

If a DEFAULT property is used for the field and the INPUT dialog does not use the WITHOUT DEFAULTS option, the runtime system assumes that the default value satisfies the `notNull` property.

Unlike the REQUIRED property which has no effect when the INPUT dialog uses the WITHOUT DEFAULTS option, the `notNull` property is always checked when validating a dialog.

orientation

The `orientation` property defines whether an element displays vertically or horizontally.

Usage

The orientation property is typically used in the definition of a [RadioGroup](#) form item, to specify how radio items have to be displayed.

parentIdColumn

The `parentIdColumn` property specifies the formField that contains the identifier of the parent node of a tree node.

Usage

This property is mandatory. This property is used in the definition of a [Tree](#), see Tree Views for more details.

PER comments

All comments imported from a `per` file are grouped in the `PER comments` properties.

picture

The `picture` property specifies a character pattern for data entry in a text field, and prevents entry of values that conflict with the specified pattern.

Usage

format-string can be any combination of characters, where the characters "A", "#" and "X" have a special meaning.

- The character "A" specifies **any letter** (alpha-numeric) character at a given position.
- The character "#" specifies **any digit** character at a given position.
- The character "X" specifies **any character** at a given position.

Any character different from "A", "X" and "#" is treated as a literal. Such characters automatically appear in the field and do not have to be entered by the user.

The picture property does not require data entry into the entire field. It only requires that whatever characters are entered conform to *format-string*.

When picture specifies input formats for DATETIME and INTERVAL fields, the form compiler does not check the syntax of *format-string*, but your form will work if the syntax is correct. Any error in *format-string*, however, such as an incorrect field separator, produces a runtime error.

The typical usage for the picture property is for (fixed-length) CHAR fields. It is not recommended to use picture for other data types, especially numeric or date/time fields: The current value of the field must always match (i.e. be formatted according to) picture.

Understand that the picture property defines a mask for data entry. In order to format fields when data is displayed to the field, use the FORMAT property instead. FORMAT is typically used for numeric and date fields, while picture is typically used for formatted character string fields requiring input control.

posX, posY

The `posX` and `posY` properties define the position of the upper left corner of the form item on the x and y axis of the form.

program

The `program` property can specify an external application program to work with screen fields of data type TEXT or BYTE.

queryEditable

The `queryEditable` property makes a combobox field editable during a CONSTRUCT statement.

Usage

The `queryEditable` property is effective only during a CONSTRUCT statement.

This property is useful when the display values match the real values in the [items](#) property.

required

The `required` property forces the user to modify the content of a field during an INPUT or INPUT ARRAY statement.

Usage

This property forces the user to modify the content of the field during the dialog execution. If the user subsequently erases the entry during the same input, the runtime system considers the `required` property satisfied.

The `required` property is effective only when the field name appears in the list of screen fields of an INPUT or INPUT ARRAY statement.

If a default property is used for the field and the input dialog does not use the `WITHOUT DEFAULTS` option, the runtime system assumes that the default value satisfies the `required` property.

If both the `required` and default properties are assigned to the same field, the runtime system assumes that the default value satisfies the `required` property.

If the dialog instruction uses the `WITHOUT DEFAULTS` clause, the current value of the variable linked to the `required` field is considered as a default value; the runtime system assumes that the field satisfies the `required` property, even if the variable value is NULL. Note however that in an INPUT ARRAY dialog, the `required` property applies always to new created rows, even if `WITHOUT DEFAULTS` is used.

To insist on a non-null entry, use the `NOT NULL` property instead. The `NOT NULL` property can be specified in the field definition or in the corresponding column in the database schema file.

reverse

On character terminals, the `reverse` property displays any value in the field in reverse video (dark characters in a bright field).

rowHeight

The `rowHeight` property forces a specific height for all rows in a Table container.

sample

The `sample` property defines the text to be used to compute the width of a `formField`.

Usage

By default, `formFields` are rendered by the client with a size determined by the current font and the number of characters used in the layout grid. The field width is computed so that the largest value can fit in the widget.

Sometimes the default computed width is too wide for the typical values displayed in the field. For example, numeric fields usually need less space as alphanumeric fields. If the values are always smaller, you can use the `sample` property to provide a hint for the front end to compute the best width for that `formField`.

When specifying the `sample` property, you do not have to fill the sample string up to the width of the corresponding field tag: The front-ends will be able to compute a physical width by applying a ratio to fit the best visual result. For example, for a sample of 'XY' used for a field defined with 10 characters, is equivalent to specifying a sample of 'XYXYXYXYXY'.

If the `sample` property is not used, the first 6 cells are always computed with the pixel width of the 'M' character in the current font. Next cells are computed with the pixel width of the '0' (zero) character. In other words, the default sample model is 'MMMMMM000000.....', reduced to the size of the field tag in the layout:

<code>-123456789-123456789-</code>	<code>default sample</code>
<code>[f01]</code>	<code>MMMM</code>
<code>[f02]</code>	<code>MMMMMM</code>
<code>[f03]</code>	<code>MMMMMM0000000000</code>

You can define a default sample for all fields used in the form, by specifying a `DEFAULT sample` option in the `INSTRUCTIONS` section.

scroll

The `scroll` property can be used to enable horizontal scrolling in a character field with character input.

scrollbars

The `scrollbars` property can be used to specify scrollbars for a form item.

Usage

This property defines scrollbars for the form item, such as a [TextEdit](#) on page 431. Options are vertical, horizontal, or both.

sizePolicy

The `sizePolicy` property is a sizing directive to display form elements.

Usage

This property defines how the front-ends will compute the size of some form elements in grids. The `sizePolicy` applies only to leaf elements, not to containers. The default value of `sizePolicy` is `initial`.

The `sizePolicy` property is ignored for the widgets used in [Table](#) and [Tree](#) columns, because in tables, the size policy is implicitly defined by the cell as fixed (i.e. the size of the column in the form layout).

When the `sizePolicy` is `fixed`, the form elements size is exactly the one defined in the Form Specification File. The size of the element is computed from the width and height in the form grid and the font used on the front-end side.

When `sizePolicy` is `dynamic`, the size of the element grows and shrinks according to the width of the wider during the life time of the application. This can be used for `ComboBox` or `RadioGroup` fields, when the size of the widget must fit exactly to its content, which can vary during the program execution. With `sizePolicy=DYNAMIC`, `Buttons`, `Labels`, `CheckBoxes`, `Images`, and `RadioGroups` can shrink and grow all the time, while `ComboBoxes` can only grow.

When `sizePolicy` is `initial`, the size is computed the first time the element appears on the screen. Once the widget is displayed, its size is frozen. This is typically used when the size of the element must be fixed but is not known at design time (for instance, when populating a `ComboBox` item list from a database table). This option is also useful when the text of labels is unknown at design time because of Internationalization. With `sizePolicy=initial`, the behavior differs depending on the form element type. Keep in mind that after the first display, the element size will be frozen:

- `Buttons`: The size defined in the form is a minimum size. If the text is bigger, the size grows (width and height).
- `ComboBoxes`: The width defined in the form is a minimum width. If one of the items in the value list is bigger, the size grows in order for the combobox to display the largest item fully.
- `Labels`, `CheckBoxes`, and `RadioGroups` can shrink or grow. The size defined in the form is ignored. The fields are sized according to the element text.
- `Images` can shrink and grow according to the picture displayed. `Images` can use the `stretch` property, so that the size of the widget can be dependant from the parent container, overriding the `sizePolicy` property. If the width and height properties have to be used, the `sizePolicy` property must be set to `FIXED`.
- Other items such as `Edit` or widget without items like `ProgressBar` are not sensitive to the `sizePolicy` property.

The `sizePolicy` property is supported for `WebComponent` fields, however as the content and behavior is defined by the front-end, this property may have no effect. See the front-end web-component specific documentation for more details.

sliderOrientation

The `sliderOrientation` property determines the vertical or horizontal orientation of the slider.

spacing

The `spacing` property is a spacing directive to display form elements.

Usage

This property defines the global distance between two neighboring form elements. In `NORMAL` mode, the front end displays form elements consistent with the desktop spacing, which is, for example, 6 and 10 pixels on Microsoft™ Windows™ platforms. Some overcrowded forms may need to be displayed with less space between elements, to let them fit to the screen. In this case you can use the `COMPACT` mode.

By default, forms are displayed with `COMPACT` spacing.

splitter

The `splitter` property forces the container to use a splitter widget between each child element.

Usage

This property indicates that the container (typically, `HBox` and `VBox`) must have a splitter between each child element held by the container. If a container is defined with a splitter and if the children are stretchable (like `Table` or `TextEdit`), users can re-size the child elements inside the container.

step

The `step` property specifies how a value is increased or decreased in one step (by a mouse click or key up/down).

Usage

This property is typically used with form items allowing the user to change the current integer value by a mouse click like [Slider](#) and [SpinEdit](#).

stretch

The `stretch` property specifies how a widget must re-size when the parent container is re-sized.

Usage

This property is typically used with form items that can be re-sized like [Image](#) or [TextEdit](#) fields. By default such form items have a fixed width and height, but in some cases you may want to force the widget to re-size vertically, horizontally, or in both directions.

style

The `style` property specifies a style for a form element.

Usage

This property specifies a presentation style to be applied to a form element. The presentation style can define decoration properties such as a background color, a font type, and so on.

styleFile

The `styleFile` property specifies a style file to apply to a form for form previews and style dialogs.

Usage

Important: The `styleFile` attribute is only used by the Form Designer when previewing the form and for Form Designer-related style dialogs. To apply a style file with your Genero application, you must load the style file using the `ui.Interface.loadStyles` method. See the *Genero Business Development Language User Guide* for more information regarding the use of style files by Genero applications.

tabIndex

The `tabIndex` property defines the tab order for a form item.

Usage

This property can be used to define the order in which the form items are selected as the user "tabs" from field to field when the program is using the [formField order option](#).

It can also be used to define which field must get the focus when a [Page](#) is selected.

By default, form items get a tab index according to the order in which they appear on the form.

Tip: `tabIndex` can be set to zero in order to exclude the item from the tabbing list. The item can still get the focus with the mouse.

tag

The `tag` property can be used to identify the form item with a specific string.

Usage

This property is used to identify form items with a specific string. It can be queried in the program to perform specific processing.

You are free to use this property as you need. For example, you can define a numeric identifier for each field in the form in order to show context help, or group fields for specific input verification.

If you need to handle multiple data, you can format the text, for example, by using a pipe separator.

text

The `text` property defines the label associated with a form item, such as the text of a checkbox item.

title

The `title` property defines the title of a form item used in a table container.

totalRows

The `totalRows` property defines the total number of rows in a Table container.

unHidable

The `unHidable` property indicates that the element cannot be hidden or shown by the user with the context menu.

Usage

By default, a [Table](#) container allows the user to hide the columns by a right-click on the column header. Use this property to prevent the user from hiding a specific column.

unHidableColumns

The `unHidableColumns` property indicates that the columns of the table cannot be hidden or shown by the user with the context menu.

Usage

Same effect as [unHidable](#), but at the [Table](#) level, to make all columns not hideable.

unMovable

The `unMovable` property prevents the user from moving a defined column of a table.

Usage

By default, a [Table](#) container allows the user to move the columns by dragging and dropping the column header. Use this property to prevent the user from changing the order of a specific column. Typically, `unMovable` is used on at least two columns to prevent the user from changing the order of the input on these columns.

unMovableColumns

The `unMovableColumns` property prevents the user from moving columns of a table.

Usage

By default, a [Table](#) container allows the user to move the columns by dragging and dropping the column header. Use this property to prevent the user from changing the order of columns.

unSizable

The `unSizable` property prevents the user from resizing the element.

Usage

By default, a [Table](#) container allows the user to re-size the columns by a drag-click on the column header. Use this property to prevent a re-size on a specific column.

unSizableColumns

The `unSizableColumns` property prevents the user from resizing columns of the table.

Usage

By default, a [Table](#) container allows the user to size the columns. Use this property to prevent the user from sizing any of the columns in the table.

unSortable

The `unSortable` property prevents the user from sorting on a specific column.

Usage

By default, a [Table](#) container allows the user to sort the columns by a left-click on the column header. Use this property to prevent a sort on a specific column.

unSortableColumns

The `unSortableColumns` property prevents the user from selecting any column of the table for sorting.

Usage

By default, a [Table](#) container allows the user to sort on the columns. Use this property to prevent the user from sorting on any columns in the table.

validate

The `validate` property is an Action Defaults property defining the data validation level for a given action.

Usage

The action default property `VALIDATE = NO` indicates that no data validation must occur for this action. However, current input buffer contains the text modified by the user before triggering the action.

Validate Like

The `Validate Like` property instructs the form compiler to set the field properties that are defined in the `.val` database schema file for the specified column.

Usage

Specifying the `Validate Like` property is equivalent to writing in the field definition all the properties that are assigned to `table.column` in the `.val` database schema file generated from the **syscolval** table.

Note that `.val` properties are taken automatically from the schema file if the field is linked to `table.column` in the field name specification. The `Validate Like` property is usually specified for `FORMONLY` fields.

The `Validate Like` property is evaluated at compile time, not at runtime. If the database schema file changes, you should recompile all your forms.

Even if all of the fields in the form are `FORMONLY`, the `Validate Like` property requires the form compiler to access the database schema file that contains the description of `table.column`.

valueChecked

The `valueChecked` property defines the value associated with a checkbox item when it is checked.

Usage

This property is used in conjunction with the [valueUnchecked](#) property to define the values corresponding to the states of a `CHECKBOX`.

This property is not used by the runtime system to validate the field, you must use the [include](#) property to control value boundaries.

See [CheckBox](#) for more details.

valueMax

The `valueMax` property defines a upper limit of values displayed in widgets (such as progress bars).

Usage

This property is typically used in [ProgressBar](#), [SpinEdit](#), [Slider](#) fields to define the upper limit.

This property is not used by the runtime system to validate the field, you must use the [include](#) property to control value boundaries.

valueMin

The `valueMin` property defines a lower limit of values displayed in widgets (such as progress bars).

Usage

This property is typically used in [ProgressBar](#), [SpinEdit](#), [Slider](#) fields to define the lower limit.

This property is not used by the runtime system to validate the field, you must use the [include](#) property to control value boundaries.

valueUnchecked

The `valueUnchecked` property defines the value associated with a checkbox item when it is not checked.

Usage

This property is used in conjunction with the [valueChecked](#) property to define the values corresponding to the states of a `CHECKBOX`.

This property is not used by the runtime system to validate the field, you must use the [include](#) property to control value boundaries.

See [CheckBox](#) for more details.

verify

The `verify` property requires users to enter data in the field twice to reduce the probability of erroneous data entry.

Usage

This property supplies an additional step in data entry to ensure the integrity of your data. After the user enters a value into a `verify` field and presses RETURN, the runtime system erases the field and requests reentry of the value. The user must enter exactly the same data each time, character for character: 15000 is not exactly the same as 15000.00.

The `VERIFY` property takes effect in `INPUT` or `INPUT ARRAY` instructions only, it has no effect on `CONSTRUCT` statements.

version

The `version` property is used to specify a user version string for the form.

Usage

This property specifies a version string to distinguish different versions of a form. You can specify an explicit version string or use the `TIMESTAMP` keyword to force the form compiler to write a timestamp string into the `42f` file.

Typical usage is to specify a version of the form to indicate if the form content has changed. This property is used by the front-end to distinguish different form versions and to avoid reloading window/form settings into a new version of a form.

You should use the `TIMESTAMP` only during development.

wantFixedPageSize

The `wantFixedPageSize` property gives a fixed height to a `Table` container.

Usage

By default, the height of a `Table` container is re-sizeable. Use this property to freeze the number of rows to the number of screen lines defined by the form design.

wantNoReturns

The `wantNoReturns` property forces a text field to reject newline characters when the user presses the RETURN key.

Usage

By default, text fields like `TextEdit` on page 431 insert a newline (ASCII 10) character in the text when the user presses the RETURN key. As the RETURN key is typically used to fire the *accept* action to validate the dialog, you can force the field to reject RETURN keys with this property.

The user can still enter newline characters with Shift-RETURN or Control-RETURN, if these keys are not bound to actions.

wantTabs

The `wantTabs` property forces a text field to insert TAB characters in the text when the user presses the TAB key.

Usage

By default, text fields like `TextEdit` on page 431 do not insert a TAB character in the text when the user presses the TAB key, since the TAB key is used to move to the next field. You can force the field to use TAB keys with this property.

The user can still jump out of the field with Shift-TAB, if this key is not bound to an action.

windowStyle

The `windowStyle` property defines the style to be used by the parent window of a form.

Usage

The `windowStyle` property can be used to specify the style of the parent window that will hold the form. This property is specific to the form. Do not confuse with the `STYLE` property, which is used to specify decoration style of the form elements.

When a form is loaded by the OPEN WINDOW or DISPLAY FORM instructions, the runtime system automatically assigns the `windowStyle` to the `style` property of the parent window element.

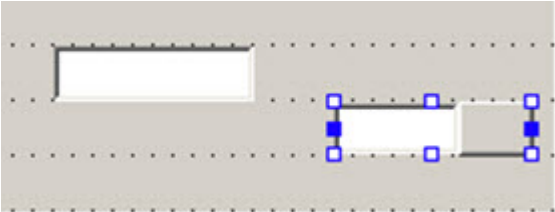
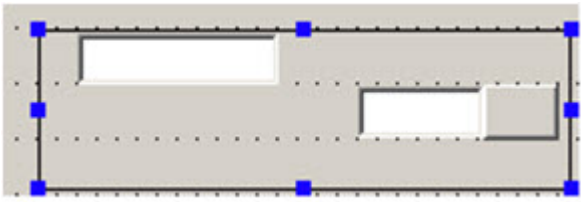
Form Designer error messages

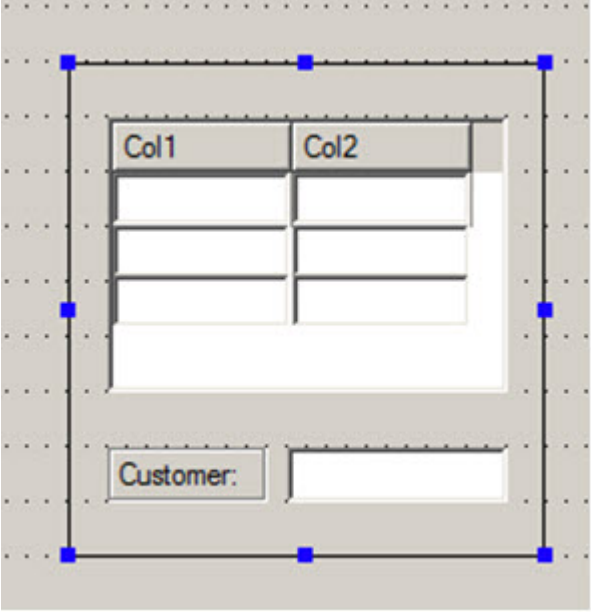
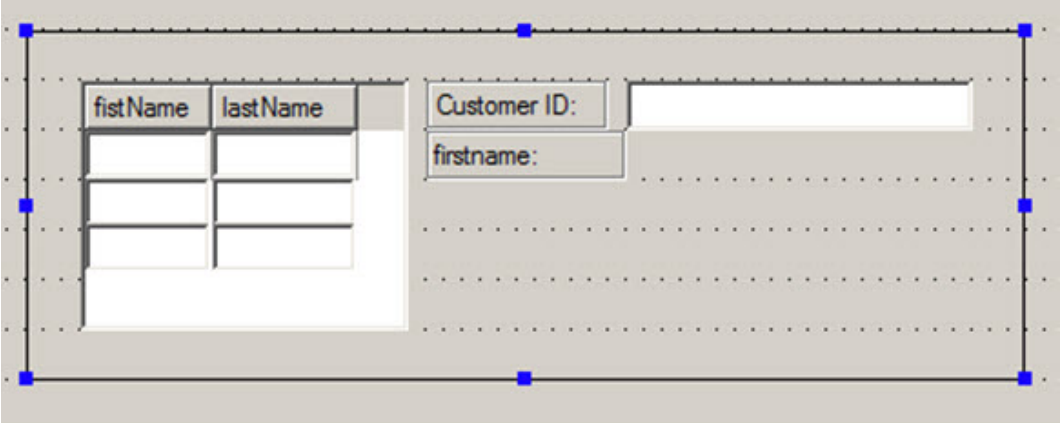
A list of Form Designer error messages. For messages that are not self-explanatory, additional information is provided.

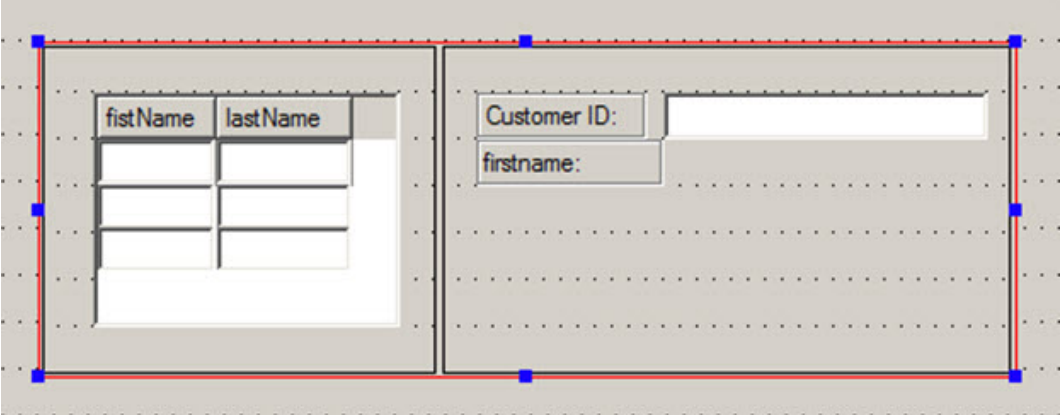
Table 147: Form Designer Error messages

Number	Description
GS-10001	Field widget used when no database is attached to form. The current form does not use a database. The Field element is an abstract Formfield that can exist only when a database is attached to a form (the widgets depends on the database column datatype).

Number	Description
	<ul style="list-style-type: none"> Change the form databaseName property to the corresponding database, Change the Formfield widget from Field to another widget (Edit, for example).
GS-10002	<p>Property value %1 not available, upgrade to genero version %2 or later.</p> <p>The value is not supported in this version of Genero.</p> <p>Change the property value or upgrade Genero.</p>
GS-10003	<p>Columns with 'aggregate' must have a widget of type 'Edit'.</p> <p>The column using an aggregate is not an 'Edit' widget type.</p> <ul style="list-style-type: none"> Change the widget type to 'Edit'. Remove the aggregate.
GS-10004	<p>Error due to fgl not set well.</p> <p>When using <code>gsform</code>, the compiler is not working correctly.</p> <ul style="list-style-type: none"> Check whether FGLDIR is set. Check whether BDL is licensed and working.
GS-10005	<p>Property %1, does not exist in its <code>wcsettings</code> in the WebComponent directory.</p> <p>Open the <code>wcsettings</code> file in Code Editor and add the missing property then reload the directory and the form file.</p>
GS-10006	<p>Invalid WebComponent directory.</p> <ul style="list-style-type: none"> Check if the WebComponent directory is blank. Check if the directory exists on disk. Check if there are any <code>.wcsettings</code> files present in the directory.
GS-10007	<p>Property %1 cannot contain '.'. Please upgrade to genero version %2 or later.</p>
GS-10008	<p>Children are out of container bounds.</p> <p>The children of the parent are out of its boundaries.</p> <ul style="list-style-type: none"> Increase the size of the parent Decrease the size of the children
GS-10011	<p>Widget shares border or intersects with another widget.</p> <p>Widgets share border or overlap with each other.</p> <ul style="list-style-type: none"> Move them so they no longer intersect or are adjacent. Delete them.
GS-10012	<p>Invalid widget, upgrade Genero version.</p> <p>The SpinEdit, TimeEdit, Slider and Field widgets were not available prior to Genero version 2.00. The widget WebComponent was introduced in Genero version 2.30. The Tree container and Phantom widget were introduced in Genero version 2.20</p> <ul style="list-style-type: none"> Change version to corresponding valid version or later.

Number	Description
	<ul style="list-style-type: none"> Change widgets to existing widgets with respect to your current version (Edit, for example), or do not use these widgets.
GS-10013	<p>Invalid property, upgrade Genero version.</p> <p>The unitWidth/unitHeight property was added in Genero version 2.00. The doubleClick/splitter/minWidth/minHeight properties were added in Genero version 2.10The image/contextMenu properties were added in Genero version 2.20. The valueMin/valueMax properties were added in Genero version 2.21. The justify/style properties were added in Genero version 2.30.</p> <ul style="list-style-type: none"> Reset to a supported property. Change version to corresponding valid version or later.
GS-10016	<p>Layout required for containers.</p> <p>Cannot have two or more containers without a layout in a parent container, except for layout tags (containers in grids).</p> <p>Layout the containers vertically or horizontally.</p>
GS-10017	<p>Nested grid/scrollgrid cannot contain containers.</p> <p>Nested grid / ScrollGrid (layout tag ScrollGrid) cannot contain these children: Table, Group, ScrollGrid, GridNested Group /Scrollgrid (layout tag Group) cannot contain containers other than: HRec, Table, Tree</p> <p>Re-arrange your elements using layouts.</p>
GS-10019	<p>Parent container required.</p> <p>A container is required between the form and the element:</p>  <p>Figure 277: Form without container</p> <ul style="list-style-type: none"> Draw a grid or scrollgrid around the element. Move the element into a Grid, Group, scrollGrid.  <p>Figure 278: Form with container</p>
GS-10020	<p>Character not supported in '%1' encoding.</p> <ul style="list-style-type: none"> Change the characters to a supported one. Change the encoding.

Number	Description
GS-10021	<p>No element below layout tag before Genero 2.01.</p>  <p>Figure 279: Form with element inside grid below layout tag</p> <p>You cannot have elements inside the grid below a layout tag (container inside grid.) Move the elements to another grid and layout both grids vertically.</p>
GS-10022	<p>Layout tag width is name length plus 4.</p> <ul style="list-style-type: none"> • Increase layout tag width. • Shorten name attribute.
GS-10026	<p>Non container widgets not beside table in grid before Genero 2.10. Cannot have widget beside layout tag table, this feature has been implemented in version 2.10.</p>  <p>Figure 280: Form with widget beside layout tag table</p> <ul style="list-style-type: none"> • Change Genero configuration fgl to 2.10 or later • Move the widgets into another grid and layout both grids horizontally

Number	Description
	 <p>Figure 281: Form with widgets into another grid and layout both grids horizontally</p>
GS-10027	<p>Empty container.</p> <p>This node is missing its required children. Item is missing in the <i>TopMenu</i> or <i>TopMenuGroup</i> or <i>Toolbar</i> or <i>ActionDefaultList</i>. Widget is missing in group, hrec etc</p> <ul style="list-style-type: none"> • Add an valid child. • Remove the container.
GS-10028	<p>ActionDefault %1 doesn't contain modified attributes applicable to current Genero version.</p> <p>ActionDefaultGroup is defined without any attribute applicable to current Genero version.</p> <ul style="list-style-type: none"> • Delete it. • Set one of these attributes: title, image, accelerators ... • The contextmenu attribute is available since version Genero 2.20, but not before that. • The validate attribute is available since Genero 2.1x, not before that. • The Accelerator4 is available since Genero 2.00, not before that.
GS-10029	<p>Redundant table records.</p> <p>There are two or more records of the table container that are exactly the same, i.e. they have the same order of recordFields.</p> <ul style="list-style-type: none"> • Reorder the recordField(s) in record that is duplicate. • Delete the duplicate record.
GS-10031	<p>Left edge cannot be shared with table right edge.</p> <p>Widget left edge cannot be shared with table right edge. For any Genero version, Genero form compiler needs a space to the right of the table.</p> <p>Enter a space to the right of the table.</p>
GS-10033	Requires minimum location.
GS-10034	Requires minimum size.
GS-10035	<p>Old widget '%1' was removed during import.</p> <p>An old widget (canvas) that is no longer supported was present in the imported form and removed from the document.</p> <p>Create another element for this widget, or ignore the error.</p>

Number	Description
GS-10036	<p>Old widget %1 was transformed to %2 during import.</p> <p>An old widget that is no longer supported was present in the imported form. It's been transformed into the form element.</p> <p>Check that the transformation was correct, or fix it (some attributes may be incorrectly set).</p>
GS-10040	<p>Expected ValueChecked to be different from valueUnChecked.</p> <p>Checkbox valueChecked and valueUnChecked attributes cannot have the same value.</p> <p>Change one of the attribute values.</p>
GS-10042	<p>Requested operation cannot be performed, please check the Genero configuration settings.</p> <p>The Genero configuration is not valid.</p> <p>Check Genero license, directory, and Genero Desktop Client.</p>
GS-10044	<p>Internal error.</p> <p>An unexpected internal error occurred during compilation.</p> <p>Contact your local support Center.</p>
GS-10045	<p>Could not delete temporary per file.</p> <p>A temporary per file cannot be deleted during the compilation process, probably because the user opened the file or changed the access rights.</p> <p>Close the temporary file, or set the access rights to read write.</p>
GS-10046	<p>Cannot generate unique tagName, gridwidth is too small.</p> <p>All available field identifiers are used during per file generation.</p> <p>Enlarge some field's gridwidth.</p>
GS-10047	<p>Unable to create per file during form compilation.</p>
GS-10049	<p>Value not compatible with data Type.</p> <p>The Value is not compatible with data type set. This error occurs when defaultValue and include property value are incompatible with dataType property.</p> <p>Change defaultValue or include; enter a value compatible to data type format.</p>
GS-10052	<p>Record cannot start with phantom field.</p> <p>Rearrange record fields.</p>
GS-10054	<p>idColumn/parentIdColumn must be defined for Tree.</p> <p>Set a valid value, i.e. a name of one of the tree's children, for these properties.</p>
GS-10055	<p>Invalid value set for %1 property.</p> <p>Check the documentation for the property, and set a valid value. For example, a valid value for the name property is comprised of alphanumeric characters and underscores, but can not start with a number:</p>

Number	Description
	<ul style="list-style-type: none"> • For accelerator, accelerator2, accelerator3, accelerator4; value should be a valid accelerator name. • For imageColumn, idColumn, parentIdColumn, expandedColumn, isNodeColumn; value should be one of the children of the table/tree. • For expandedColumn, isNodeColumn, imageColumn; value could be blank, i.e. reset to default. • For Action; value should be a valid action name. • For FieldType; the Field widget cannot be used with FORM_ONLY or FORM_LIKE fieldType; the expected fieldType is TABLE_COLUMN or TABLE_ALIAS. • For formfields; value should be <i>tableName.colName</i>. • For formonly formfields; value should be <i>formonly.name</i>. • For tableAlias type formfields; value should be <i>tableAliasName.colName</i>. • For non-formfields; value should be a valid name.
GS-10056	<p>Include property doesn't support dataType.</p> <p>An include string is set while dataType is not supported (for example, interval).</p> <p>Change data type, or remove include string.</p>
GS-10057	<p>Value is out of data Type range.</p> <p>Some data types have a limitation of size (for example, integer).</p> <p>Change data type (for example, from integer to bigint), or modify value so that it is valid.</p>
GS-10061	<p>Screen record array has different component sizes.</p> <p>Matrices corresponding to the Screen Record need to have the same repeat count (number of rows * number of columns).</p> <ul style="list-style-type: none"> • Change the row count or the column count. • If the current screen record is an array and any of the screenRecordItems correspond to a FormField, convert it to a matrix. • If the current screen record is not an array, and any of the screenRecordItems correspond to a Matrix, convert it to a FormField.
GS-10062	<p>All members of '\Record\' must reference the same container.</p> <p>The formfields corresponding to the record field in the same record must belong to the same container.</p> <ul style="list-style-type: none"> • Move the record field to the correct record. • Move the formfield to the right container. • Remove the record field or formfield.
GS-10063	<p>First tree column must have a widget of type Edit or Label.</p> <p>The first Tree widget column is not an Edit or a Label.</p> <ul style="list-style-type: none"> • Insert an Edit or Label as the first column. • Change the first column to an Edit or Label. • Re-order the columns to have an Edit or Label as the first column.
GS-10064	<p>Invalid/Duplicate name transformed during import.</p> <p>An invalid name that was present in the imported form was transformed to a valid and unique name. A valid name is not blank, and is made up of these characters: A-Z, 0-9, "_". You may edit the transformed name.</p>

Number	Description
GS-10065	<p>Multiple lines in Form title.</p> <p>Genero clients don't support multiple lines.</p> <p>Remove the carriage return.</p>
GS-10070	<p>Cannot save 4fd file.</p> <p>The 4fd file cannot be saved due to file system permissions.</p> <p>Check permissions, or move the per file and import it again.</p>
GS-10076	<p>File not found.</p> <p>A file is missing.</p>
GS-10077	<p>Cannot load file.</p> <p>An unexpected error occurred loading the file.</p>
GS-10079	<p>Cannot import file, %1 already exists.</p> <p>The form file already exists, import is canceled (gsform).</p> <ul style="list-style-type: none"> • Delete the 4fd form file • Force overwrite (-i option).
GS-10081	<p>Container is too small, user defined properties will be ignored.</p> <p>The layout tagged container is too small to hold identifier. Any user defined properties will be ignored. This is a warning message.</p> <p>Increase width of container.</p>
GS-10082	<p>Record has only phantom fields.</p> <ul style="list-style-type: none"> • Change widget type of a record field Formfield from phantom to some other type. • Drag-drop some record field that is not a phantom type into this record. • Delete record.
GS-10083	<p>valueMin must be lower than valueMax.</p> <ul style="list-style-type: none"> • Change valueMin so that it is less than valueMax. • Change valueMax so that it is greater than valueMin.
GS-10084	<p>Invalid text.</p> <p>The text property value cannot contain newline characters, '\n' or '\r'.</p> <p>Delete or replace these characters.</p>
GS-10085	<p>Wrong file extension.</p> <p>The file chosen for import has an incorrect extension.</p> <p>Select a file with a per extension to import.</p>
GS-10086	<p>Cannot create compilation task.</p> <p>Check the build rule; if correct, contact your FourJ's Support center.</p>
GS-10087	<p>Invalid widget position.</p>

Number	Description
	Change position of widget.
GS-10088	<p>StepX needs to be greater than 1.</p> <p>The property stepX must have a value greater than 1.</p> <ul style="list-style-type: none"> • Edit stepX property. • Uncheck repeat property.
GS-10089	<p>Invalid nested table position.</p> <p>Change table position.</p>
GS-10090	<p>Invalid file, closing tag does not correspond to the opening one.</p> <p>The file is invalid; it contains mismatching tags and XML parsing is not possible.</p>
GS-10091	Invalid file, orphan closing tag.
GS-10092	<p>Empty space required on all sides of layout tag.</p> <p>Minimum of one unit empty space is required on all sides of layout tag containers.</p> <p>Re-size layout tag container or its children.</p>
GS-10095	<p>componentType property must be defined as a name of valid webComponent.</p> <p>Set a valid value from available WebComponents.</p>
GS-10097	<p>Invalid WebComponent.</p> <p>The WebComponent is invalid as it does not satisfy XSD file rules.</p> <p>Open wcsettings file in code editor and rectify in accordance with XSD validation.</p>
GS-10098	<p>Invalid WebComponent schema.</p> <p>The WebComponent XSD file that is used for validating WebComponent XML files is invalid.</p> <ul style="list-style-type: none"> • Take a clean copy from of file. • Undo all changes done manually.
GS-10099	<p>Empty text widget will be ignored.</p> <p>The text property value will be ignored as it is blank. This is a warning message.</p> <p><i>Solution:</i> Since property value is blank, it plays no role, and is ignored.</p>
GS-10100	<p>Property '%1' conflicts with 'id' or 'parentid'.</p> <p>The value of property 'expandedColumn' or 'isNodeColumn' is same as either 'idColumn' or 'parentIdColumn' of tree widget.</p> <p>Change the property value of either of the properties so that expandedColumn and isNodeColumn do not have the same value as either idColumn or parentIdColumn.</p>
GS-10300	Need one and only one relation for program %1.

Business Records error messages

A list of Business Records error messages. For messages that are not self-explanatory, additional information is provided.

Table 148: Business Records Error Messages

Number	Description
GS-24001	<p>Error loading file <i>file</i>.</p> <p>An error occurred loading the file.</p> <p>Check the file name and permissions .</p>
GS-24002	<p>Malformed XML.</p> <p>XML file content is invalid.</p> <p>Select the correct file or correct the XML.</p>
GS-24003	<p>File <i>file</i> not found in Business Application diagram (<BA file name>).</p> <p>A referenced element is not found in the Business Application diagram.</p> <p>Select the correct file.</p>
GS-24004	<p>Invalid value for %1 property.</p> <p>Check the property syntax in the documentation and correct any errors.</p>
GS-24005	<p>Conflicted item.</p> <p>The Business Application diagram and Business Record unique id do not match.</p> <p>Open the 4ba file and resolve the conflict.</p>
GS-24201	<p>Missing master Record.</p> <p>The document must contain at least one record that is master.</p> <p>Create a record.</p>
GS-24202	<p>Unique query key must be set.</p> <p>At least one field must be declared the unique key in a record.</p> <p>Set a unique key.</p>
GS-24203	<p>Record used in relation must be active.</p> <p>Change the record to active, or remove the relation.</p>
GS-24204	<p>Empty relation.</p> <p>This relation has no field definition.</p> <p>Add foreign/primary (or source/destination) fields.</p>
GS-24205	<p>Database table column referenced more than once in the form.</p> <p>In one form, a database table column can be referred only once. Table aliases should be used to attach more than one field to a database.</p> <ul style="list-style-type: none"> Select one of the wrong Formfields and change the fieldType attribute from table_column to table_alias.

Number	Description
	<ul style="list-style-type: none"> Select one of the wrong Formfields and change fieldType attribute to non_database.
GS-24206	<p>Missing master table for <i>record</i> record.</p> <p>Set the master table property.</p>
GS-24207	<p>Table %1 unused in query.</p> <p>A database table is used in the record, but there is no join for it in the query.</p> <p>Add a join for the table in the record query.</p>
GS-24208	<p>No schema attached.</p> <p>A database is required for the business records.</p> <p>Set the database name property to an existing database.</p>
GS-24209	<p>Invalid relation, <i>primaryField</i> and <i>foreignField</i> must have same number of fields.</p> <p>Modify the relation fields so that a primary field corresponds to each foreign field.</p>
GS-24210	<p>Invalid query, left and right join must have same number of columns.</p> <p>Check the query fields so that a left field corresponds to each right field.</p>
GS-24211	<p>Relation types don't match exactly.</p> <p>The type of a foreign key doesn't match the corresponding primary key's type.</p> <p>Check the relation, change the field types, or fix the primary / foreign key.</p>
GS-24212	<p>Relation field <i>fieldname</i> not found.</p> <p>The relation refers a field that is missing in the record.</p> <p>Modify the relation or add the field to the record.</p>
GS-24213	<p>Non existing schema <i>schema</i> attached to document.</p> <p>Check the schema being referenced.</p>
GS-24214	<p>Nonexistent table <i>table</i> referenced in document.</p> <p>The document uses a database table that is not present in the schema.</p> <p>Update the schema, or change the field using the table.</p>
GS-24215	<p>Nonexistent column <i>table.column</i> referenced in document.</p> <p>The document uses a database column that is not present in the schema.</p> <p>Update the schema or change the field using the <i>table.column</i>.</p>
GS-24216	<p>Database table defined in "no database" document.</p> <p>Set the database property to an existing schema or remove the field.</p>
GS-24217	<p>Table alias referenced more than once in the form.</p> <p>The same table alias is associated with two columns in the form.</p> <ul style="list-style-type: none"> Change one table alias name.

Number	Description
	<ul style="list-style-type: none"> Change either the table or column name.
GS-24218	<p>Alias <i>alias</i> referenced for different tables in document.</p> <p>The same alias is used for different tables.</p> <p>Rename the alias so that it refers to the same database table.</p>
GS-24219	<p>Alias Lookup field is ignored on %1 as it is the master table of the record.</p> <p>A lookup field is dedicated to foreign field update in the master table; do not set it on a master table column.</p> <p>Remove the lookup property value or update the field database settings.</p>
GS-24220	<p>Name value %1 is already used.</p> <p>Duplicate name used in the document.</p> <p>Rename the element so that the name is unique.</p>
GS-24221	<p>Invalid INTERVAL qualifier.</p> <p>The <i>qual1</i> or <i>qual2</i> set for the INTERVAL <i>sqlType</i> is not valid. It should belong to the INTERVAL classes (i.e., YEAR-MONTH or DAY-TIME)</p> <p>Change either <i>qual1</i> or <i>qual2</i> to fit the respective class range or change the <i>sqlType</i> property from INTERVAL to another <i>sqlType</i>.</p>
GS-24222	<p>Startfield of DATETIME or INTERVAL qualifiers must come earlier in the time-list than its endfield.</p> <p>The <i>qual1</i> value should be greater than the <i>qual2</i> value, when the <i>sqlType</i> is either DATETIME or INTERVAL.</p> <p>Ensure <i>qual1</i> is greater than <i>qual2</i> or change the <i>sqlType</i> property from INTERVAL to another <i>sqlType</i>.</p>
GS-24223	<p>Query properties set without attached database schema.</p> <p>Some query properties (join, order, additional tables or where) are defined without a database schema attached.</p> <p>Clear the query properties or attach a database schema to document.</p>
GS-24224	<p>Table <i>table</i> referenced by <i>property</i> is not present in the record.</p> <p>A table is referenced in a query property (join or order), but is not referenced in any record's field or additional tables.</p> <ul style="list-style-type: none"> Remove the join or order that references this table. Add a field referencing a column from this table to the record. Add the table to the 'additional properties' property. In the join or order, change the table to another one that is present in the record.
GS-24225	<p>Invalid relation, it must have at least one field.</p> <ul style="list-style-type: none"> Remove the relation. Add one or more source, destination field pair.

Number	Description
GS-24226	<p>Value not compatible with dataType.</p> <p>The value is not compatible with dataType set. This error occurs when <i>defaultValue</i> and <i>include</i> property value is incompatible with dataType property.</p> <p>Change <i>defaultValue/include</i>, enter a value compatible to dataType format.</p>
GS-24227	<p>Invalid join between %1 and %2.</p> <p>A join between the two tables is not correct (another one exists with a different operator).</p> <ul style="list-style-type: none"> • Remove the join • Change the operator • Change the join table(s)
GS-24228	<p>There is no join between table %1 and master table.</p> <p>The specified table is joined to another one, but not to the master table, creating a Cartesian product.</p> <ul style="list-style-type: none"> • Add the missing join for the %1 table • Remove the table %1
GS-24229	<p>Invalid initializer.</p> <p>The initializer format is incorrect.</p> <p>Change the initializer to respect the format.</p>
GS-24230	<p>Invalid source %1 for initializer %2.</p> <p>The initializer source (left of “:”) is unknown.</p> <p>Change the initializer to respect the format.</p>
GS-24231	<p>Initializer property %1 is missing.</p> <p>The property used in the initializer does not exist.</p> <p>Change the property name in the initializer.</p>
GS-24232	<p>Cannot resolve initializer value.</p> <p>The initializer cannot be resolved, the database element is not found, or the property is missing.</p> <p>Change the initializer to point to a valid element.</p>
GS-24233	<p>Orphan property %1, clean document settings to remove it.</p> <p>The document contains a dynamic property which is not present in the Application Generator template directory settings (orphan property). These orphan properties won't be taken into account during compilation. The Application Generator settings dynamic properties should match the document ones, otherwise this error is generated.</p> <ul style="list-style-type: none"> • Add the orphan properties to the <code>settings.agconf</code> file. • Remove the properties from the document using Tools >> Specific setup >> Clean orphan properties.
GS-24234	<p>Orphan property group %1.</p>

Number	Description
	<p>The document contains a dynamic property group which is not present in the Application Generator template directory settings (orphan property). The Application Generator settings dynamic properties should match the document ones, otherwise this error is generated.</p> <ul style="list-style-type: none"> • Add the orphan property groups to the settings.agconf file. • Remove the properties from the document using Tools >> Specific setup >> Clean orphan properties.
GS-24235	<p>File type not defined in Application Generator settings.</p> <p>A file used in Application Generator to generate code (\$generate) is of an undefined file type in settings.agconf.</p> <p>Add the item type definition to the settings.agconf.</p>
GS-24236	<p>Cannot resolve initializer without a valid database schema.</p> <p>The property initializer uses the database schema but the file doesn't have a schema.</p> <ul style="list-style-type: none"> • Set the database. • Set the property value so that the initializer is not resolved. • Remove the initializer.
GS-24237	<p>Unique key field %1 is not present in the Record.</p> <p>The unique key field value contains one field which is not in the record.</p> <ul style="list-style-type: none"> • Remove the field from the unique key. • Add the field to the record.
GS-24238	<p>Node %1 contains orphan properties, clean document settings to remove them.</p> <p>The file settings for the node differ from the Genero Studio ones, either the Genero Studio settings are not up to date, or the file contains old settings.</p> <ul style="list-style-type: none"> • Update the Genero Studio settings. • Clean the document settings (Tools >> Specific setup >> Clean orphan properties).
GS-24239	<p>Inactive table %1 in the database schema.</p> <p>The database table active flag is set to false, the table is inactive and cannot be used for code generation.</p> <p>Make the table active or do not use it.</p>

XML validation error messages

A list of XML validation error messages. For messages that are not self-explanatory, additional information is provided.

Table 149: XML Validation Error messages

Number	Description
GS-10400	Invalid root element.

Number	Description
	Edit the root to be <ManagedForm>.
GS-10401	<p>Invalid child %1.</p> <p>The element is not a valid child of the parent.</p> <p>Edit the child to be an acceptable child of the parent.</p>
GS-10402	<p>%1 occurs more than once.</p> <p>An element is defined more than once, eg: Topmenu, Toolbar can appear just once.</p> <p>Remove the duplicate occurrences to keep only a single valid definition.</p>
GS-10403	<p>Property %1 occurs more than once.</p> <p>A property for an element is defined more than once.</p> <p>Remove the duplicate occurrences to keep only a single valid definition.</p>
GS-10404	<p>Unexpected property %1 is ignored.</p> <p>This warning is displayed when a property that does not belong to the element is defined. This is ignored.</p> <p>Remove the property that does not belong.</p>
GS-10405	<p>Incomplete %1 definition, missing property(s) %2.</p> <p>Some mandatory property or properties of an element have not been defined.</p> <p>Contact your support center - provide the 4fd file, the Genero Studio version, and how this file was created.</p>
GS-10407	<p>Invalid recordField, corresponding formfield not found.</p> <p>The recordField does not correspond to any formfield, i.e. no formfield is present whose fieldId property matches the fieldIdRef property of recordField.</p> <ul style="list-style-type: none"> • Check if recordField definition is invalid, and make it valid. • Add a valid formField. • Remove the recordField.
GS-10408	<p>Incomplete Table definition for %1, missing record.</p> <p>The table does not have a corresponding record defined.</p> <ul style="list-style-type: none"> • Check if any existing records are invalid and could be table's record. • Add a valid record. • Remove table.
GS-10409	<p>Invalid record, corresponding table not found.</p> <p>The record does not correspond to any table.</p> <ul style="list-style-type: none"> • Check if record definition is invalid and make it valid. • Add a valid table. • Remove the record if not needed.
GS-10410	<p>Incomplete %1 definition, missing mandatory child.</p> <p>The element's mandatory child has not been defined.</p>

Number	Description
	Add the missing child.
GS-10411	<p>Invalid geometry for %1.</p> <p>The geometry of element is invalid.</p> <p>Edit the geometry properties <code>posX</code>, <code>posY</code>, <code>gridWidth</code>, <code>gridHeight</code> such that the element lies within the boundaries of its parent.</p>
GS-10412	<p>Incomplete FormField definition for %1, missing recordField.</p> <p>The formField does not have a corresponding recordField defined.</p> <ul style="list-style-type: none">• Check if any existing recordFields are invalid and could be corresponding to formfield.• Add a valid RecordField.• Remove FormField.
GS-10413	<p>Malformed XML.</p> <p>The xml structure is invalid.</p> <p>Check for missing tags.</p>

File Browser

File Browser is a tool to navigate, open, delete or rename files, and to create new folders or files on a file system.

File Browser manages files based on the kind of information that they contain (MIME type file management). File Browser will launch the specific action / executable defined as the default for each file type, or you may choose between the actions available for that file type.

For example, *Code Editor* opens for a `.4g1` file, and *Form Designer* for a `.4fd` file.

The File Browser displays in a view named **Files**.

Navigating files in File Browser

The folders of the file structure in File Browser are displayed as a tree.

Open the File Browser with **Tools >> File Browser** or **Windows >> Views >> Files**.

Icons indicate the folders and file types. If the file is under [Source Code Management - SVN](#) on page 529, the status is indicated on the icon.

The integrated Toolbar includes options for navigating the tree and refreshing the directory. Navigation history is maintained across sessions of Genero Studio and accessed by the drop down arrows of the previous and next buttons in the tool bar.

To show hidden files in the File Browser, check the **Show Hidden Files** box in **Tools >> Preferences, File Browser**.

Selecting files in File Browser

Click the file name to select it, double-click to open it.

Files will open in the Genero Studio module associated with it.

To select consecutive files or folders, click the first item, press and hold down SHIFT, and then click the last item. To select nonconsecutive files or folders, press and hold down CTRL, and then click each item.

Managing files in File Browser

Manage file actions such as open, cut, copy, paste, rename, and delete.

Use the mouse to drag and drop folders and files from one folder into another. Select a file listing, and right-click to display the **actions** available for that file (**Open, Cut, Copy, Paste, Rename, Delete**, and so on), based on the mime type of the file.

The **New file** option allows you to create a new document. See the [File >> New](#) on page 94 main menu option for additional information about creating new files.

You can [associate file types](#) handled by Genero Studio applications with predefined actions, and [create user-defined actions](#).

Locate a file (starting at File Browser)

From the File Browser, you can locate the file in the System File Browser.

Before you begin, the File Browser (**Files** view) is open. See [Navigating files in File Browser](#) on page 500.

You have a file visible in the **Files** view that you wish to locate in the System File Browser for your operating system.

1. Right-click on the file.
The contextual menu displays.
2. Select **Locate in System File Browser**.
The System File Browser opens, showing the location of the file.

Graphical Debugger

The *Graphical Debugger* provides a graphical interface to test and control the behavior of a Genero application. Navigate through the functions and create and manage breakpoints on functions and code lines. Choose and group together any variables to watch. Follow a number of variables easily, and even alter their values while the application is running, for testing purposes.

- [Controlling program execution](#) on page 502
- [Debugger output](#) on page 509
- [Examining data](#) on page 509
- [Examining execution flow](#) on page 509
- [Record/replay a macro](#) on page 510
- [Profiler](#) on page 510
- [Local vs. remote debug](#) on page 511
- [Reference](#) on page 511

Controlling program execution

Information on using the debugger.

- [Start the Debugger](#) on page 502
- [Stop the Debugger](#) on page 505
- [Debugger output](#) on page 509
- [Step through the program](#) on page 506
- [Breakpoints](#) on page 506
- [Watchpoints](#) on page 507

Start the Debugger

To launch the Debugger right-click on a program node and select **Debug**.

Graphical Debugger can be launched from within Project Manager:

- Right-click a program node and select **Debug** from the contextual menu; this will launch the selected application in debug mode.
- Use the **Debug** Toolbar icon; this will launch the default application.

If the Debug option is not enabled, check:

- Is the source file directory physically available (delete / rename) ?
- Are you using the correct configuration to access this project ? (local / remote) ?

Once the debugger is launched, the program begins execution. If you have [set a breakpoint](#) in a Genero file (4g1), execution will stop at that breakpoint, and the source file containing the breakpoint is opened in the Document view.

Start the Debugger on a running program

You can debug a running program.

The procedure you follow to debug a running program will depend on whether the program is local or remote.

- [Debug a running local program](#) on page 503
- [Debug a running remote program](#) on page 503

Debug a running local program

Complete this procedure to attach the Graphical Debugger to a running local process.

Before you begin:

- You have a Genero program that is running locally.

1. Select **Debug >> Attach to Process....**

The **Attach to process** dialog opens.

2. Select the **Attach to local process** radio button.

3. In the list of processes, select the process you wish to attach to.

Tip: Each process is identified by the process ID, or PID. On Windows™ systems, you can use the Task Manager to map the process ID to the `fglrun` command that started the Genero program.

4. Click **Attach**.

The debug session starts. The **Select the file** dialog opens, as the debugger needs to know where the source files are located.

5. Navigate to the directory that contains the source files for your application and select the appropriate source (`.4gl`) file, then click **Open**.

The source file opens. You are now in a standard debug session.

With the debug session open, follow the procedures for using the Graphical Debugger.

Debug a running remote program

Complete this procedure to attach the Graphical Debugger to a running remote process.

Before you begin:

- You have a Genero program that is running remotely.
- If the source files are located on a remote machine, you must mount a network drive.

1. Select **Debug >> Attach to Process....**

The **Attach to process** dialog opens.

2. Select the **Attach to remote process (SSH)** radio button.

3. Complete the required fields.

Host	The remote host.
Port	The port number for communicating with the remote host.
User	The user name needed to connect to the remote host.
Password	The password needed to connect to the remote host.
FGLDIR	The <i>FGLDIR</i> directory for the version of Genero containing the <code>fglrun</code> used to launch the program.
Process ID	The process ID of the Genero program.

4. Click **Attach**.

The debug session starts. The **Select the file** dialog opens, as the debugger needs to know where the source files are located.

5. Navigate to the directory that contains the source files for your application and select the appropriate source (`.4gl`) file, then click **Open**.

The source file opens. You are now in a standard debug session.

With the debug session open, follow the procedures for using the Graphical Debugger.

Debug a Web services server application

Complete this procedure to debug a Web services server application

Before you begin:

- You have a Genero Studio project that includes a Web services server application and (optionally) a Web services client application.
- Your Genero Studio configuration for Web applications is correctly configured, and references the correct parent application (defaultgst-debug) and parent service (ws.defaultgst-debug). These are defined for you in the default configuration for Web applications; you shouldn't need to define these yourself.

This procedure assumes you wish to debug the server side of a Web services application pair. For this procedure, you must run the client and the server application using the Web configuration.

Tip: To debug the client side of a Web services application, simply ensure that the Web services server application is running, then start the client application in debug mode.

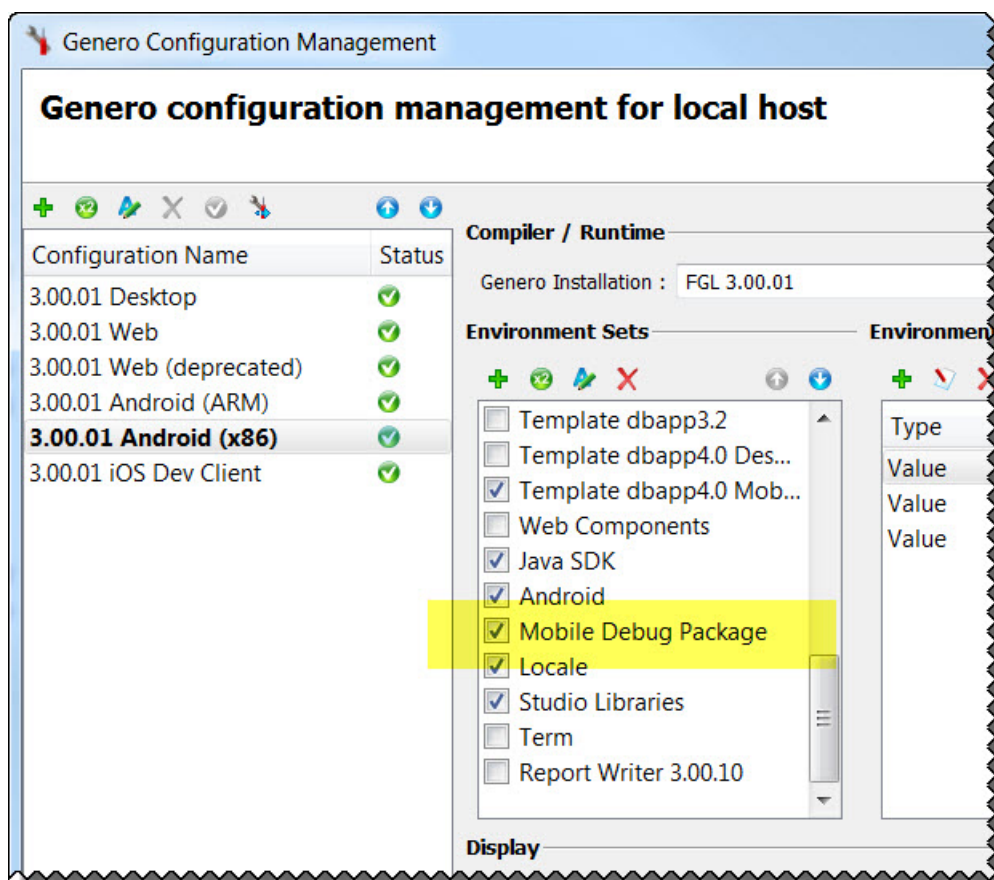
1. Select the Web configuration for Genero Studio.
2. Ensure the **Web service** property is selected for the Web services server application.
3. Add a breakpoint to the Web services server application.
4. With the server application selected in the Projects view, select **Debug >> Debug** to run the server application in debug mode.
A page opens in a Web browser using the Web service URL. This is how Genero Studio starts the Web service.
5. Start the client application.
You can either:
 - Run an external client program.
 - Start the client application from the same project as the Web services server. Right-click on the client application and choose **Execute**. A dialog appears, informing you that applications are running and asking you whether you wish to close the applications. Click **No**. A new Web page opens for the client application.
6. Use the client application. The Web services server application interrupts execution when the breakpoint is hit.
7. Use Genero Studio to debug the application.

Debug a mobile application

Complete this procedure to attach the Graphical Debugger to a running mobile process.

Before you begin:

- Genero Studio is set to use a mobile configuration.
- The mobile application was packaged and deployed with the **Mobile Debug Package** environment set selected.



See [Create a debug version of a deployed app](#) on page 919.

Figure 282: Genero Configuration Management dialog

1. Run the debug version of your deployed app.
 - For Android, see [Run the debug version of a deployed app \(Android\)](#) on page 919.
 - For iOS, see [Run a debug version of a deployed app \(iOS\)](#) on page 920
2. Select **Debug >> Attach to Mobile Process**.
This option uses your current configuration to connect to the mobile device.
3. Navigate to the directory that contains the source files for your application and select the appropriate source (.4gl) file, then click **Open**.
The source file opens. You are now in a standard debug session.

With the debug session open, follow the procedures for using the Graphical Debugger.

Stop the Debugger

What happens when you stop a debug session depends on how the debug session was started.

Regardless of how the debug session was started, the Graphical Debugger will automatically exit when the program being debugged terminates.

To exit the Debugger, select **Abort last task** from the **Debug** Menu.

- If the program was started in debug mode, the program terminates.
- If the debugger was attached to a running process, the debugger terminates. The program continues to run.

Step through the program

The `step` command allows you to "step" through your program executing one line of source code at a time. When a function call appears within the line of code, that function is also stepped through.

Once launched, program execution will be halted at the first breakpoint encountered, waiting for your action. Debugger commands can then be selected from the Toolbar, or Debug menu options.

Breakpoints

- [Set a Breakpoint](#)
- [Conditional breakpoints](#) on page 507

Set a breakpoint

To examine an area of code more closely, set a breakpoint at the desired line or function. A right-click on a line of code allows you to set, disable, and delete breakpoints.

Add/Delete breakpoint

Sets / removes a breakpoint from the current line. Clicking in the gutter will also set / remove a breakpoint at the corresponding code line.

Enable/Disable breakpoint

If checked, the current breakpoint is active; otherwise it is disabled, but still exists and can be reactivated later. Enabled breakpoints are marked by a red dot; the dot has an empty center if the breakpoint is disabled.

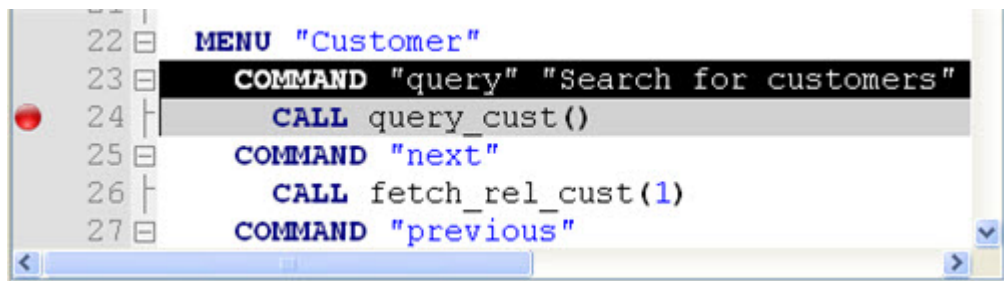


Figure 283: Set Breakpoint

About breakpoints

When the program is run, execution will stop at this line, waiting for your action. A breakpoint icon is shown in the gutter.

Breakpoints are processed as private to each user. They are linked to a source code line, not to a line number; therefore, altering a source file will not affect the breakpoint's relative position in functions.

Breakpoints are stored when the Debugger is stopped, and set when the Debugger is started. Previously set breakpoints will be available each time the Debugger is run.

Initial breakpoint

Set initial breakpoints prior to launching the Debugger. Otherwise, the application executes normally within the Debugger framework and you must use the Debugger **Interrupt** option to suspend it to monitor Debug views or employ Debugger functions. Setting a breakpoint at the first function call is usually a good starting point.

Additional breakpoints

After the initial Breakpoint is set, you can either set additional breakpoints prior to starting the Debug session or start the session with the single breakpoint. Breakpoints on Function entry points automatically

open the function source module during the debug session so you can examine the code more closely and even follow program execution line-by-line in the function if needed. You may find it helpful to set function breakpoints before starting the session, but you can also set additional breakpoints as you use Debugger commands to navigate during the session.

Tip: You can use the Code Structure view to quickly identify and open source modules so that you can set breakpoints as desired. This is especially useful for functions in external modules.

Conditional breakpoints

You can specify that the breakpoint is conditional, using a Boolean expression to determine whether to stop execution of the program.

In the **Condition** column of the **Breakpoints** tab, enter the condition or open the **Edit Expression** dialog. If a condition is defined, it will be checked each time the line is to be executed. Execution will stop at this breakpoint only if the condition is true.

Example:

```
i > 50
```

Watchpoints

- [Set a watchpoint](#) on page 507
- [Conditional watchpoints](#) on page 508

Set a watchpoint

Watchpoints can be set to stop program execution each time the value of an expression changes. In order to set the watchpoint, the program must be running in the Debugger.

1. [Set a breakpoint](#) in the function that has the variable expression in scope.
2. [Start debugging](#) the program.
3. When the program stops at the breakpoint, select the [Data view](#) on page 513 tab to display the variables.
4. Right-click the variable and select **Add to watch** to add the variable to the Watchpoints view.

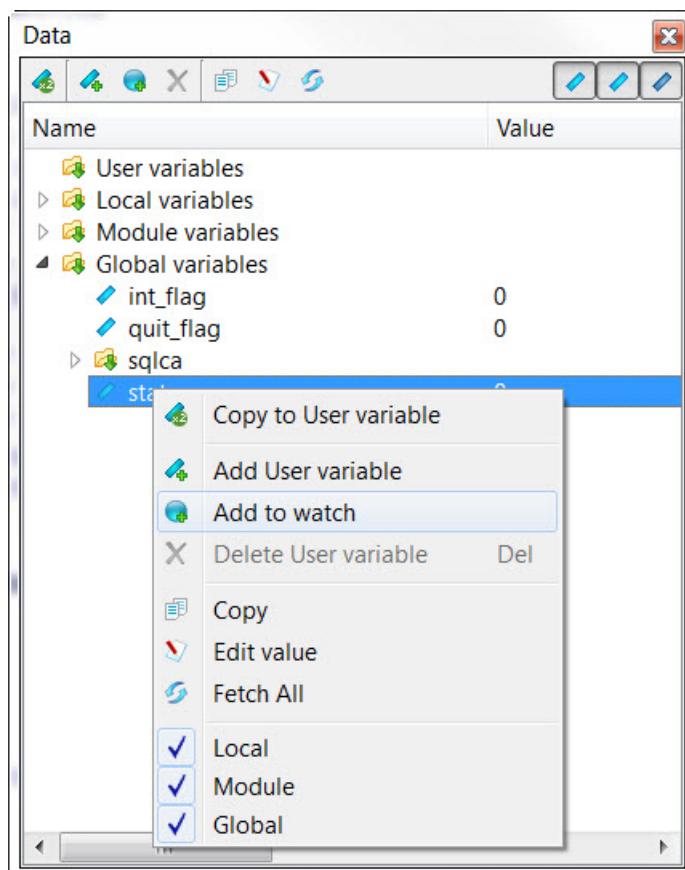


Figure 284: Setting a WatchPoint

5. Continue running the program. The program will stop when the value of a watched variable changes. You can see the values in the Watchpoint view.

Conditional watchpoints

You can specify that the watchpoint is conditional, using a Boolean expression to determine whether to stop execution of the program when the variable value has changed.

Each time the value of the watched variable changes (and the BOOLEAN expression, if used, is TRUE), the program will stop, and the variable values will be displayed Watchpoints view.

In the **Condition column** of the [Watchpoints view](#), enter the condition or press the ... button to open the Edit Expression dialog.

Or, you can enter the watch Debugger instruction and condition in the Commands view.

```
watch expression boolean-expression
```

Example:

```
watch i if i > 3
```

Debugger (fgldb) command prompt

The Graphical Debugger also provides a command line interface. Once you have started a debug session, you can enter fgldb commands directly into the fgldb **Command** view.

You can [Start the Debugger](#) on page 502 in Genero Studio and then use the command window to enter fgldb commands, or you can start a debug session at the command line (outside of Genero Studio), with the `-d` option: `fglrun -d myprog`.

Debugger output

The Debugger provides multiple views to assist you with program analysis during the Debug session.

- [Command view](#) on page 512
- [Data view](#) on page 513
- [Breakpoints view](#) on page 514
- [Watchpoints view](#) on page 514
- [Backtrace view](#) on page 515

Examining data

Monitor the values of program variables while the program is running.

Variables

The [Data view](#) on page 513 displays variable values.

Global variables

Show the variables defined with global scope as well as any Genero predefined variables used in the application, such as `INT_FLAG`.

Module variables

Show the variables defined with module scope.

Local variables

Show the variables defined with local scope for the currently executing function. Local variables are separated from argument for better visibility.

User variables

Allow you to group together variables from any of the other folders, for convenience in testing. To add a variable to the User variables set, or delete an existing User variable, right-click the variable listing to invoke the contextual menu.

User variables which are not in the scope of the current context are greyed.

Examining execution flow

Information on the call stack, stack frames, and navigating the stack.

The call stack and stack frames

Each time your program performs a function call, information about the call is saved in a block of data called a *stack frame*. Each frame contains the data associated with one call to one function.

The stack frames are allocated in a region of memory called the *call stack*. When your program is started, the stack has only one frame, that of the function `MAIN`. This is the initial frame, also known as the outermost frame. As the debugger executes your program, a new frame is made each time a function is called. When the function returns, the frame for that function call is eliminated.

The Debugger assigns numbers to all existing stack frames, starting with zero for the innermost frame, one for the frame that called it, and so on upward. These numbers do not really exist in your program; they are assigned by the Debugger to allow you to designate stack frames in commands.

Each time your program stops, the Debugger automatically selects the currently executing frame and describes it briefly. You can use the frame command to select a different frame from the current call stack.

Navigate the stack

Use the **Up/Down** options to track function calls. **Up** displays the calling function. The **Up** command advances toward the outermost frame each time it is processed. The [Data view](#) on page 513 display is based on the new context.

Down is the opposite of **Up**: it comes back to the inner frame down to the last executed line.

Record/replay a macro

You can record the Debugger commands that you are executing, save them in a file, and to execute the commands again from that file.

Options on the **Debug** menu allow you to record, stop, and play a macro.

Record macro	Start recording Debugger commands.
Stop record	Stop command recording and store the commands in a file.
Play macro	Execute the recorded commands from the file.

Profiler

The Profiler is a tool built in the runtime system that generates a report about where the program spends time, and which function calls which function. The Profiler can help to identify areas in the program that are slower than expected.

Use the **Debug >> Execute with Profiler** menu option to start the Profiler. After the program finishes executing, the Profiler information is displayed in the Output tab. The output contains the list of the functions called while the programs was running. It is presented as a five-column table.

Table 150: Flat profile columns

Column Name	Description
count	Number of calls for this function.
%total	Percentage of time spent in this function. Includes time spent in subroutines called from this function.
%child	Percentage of time spent in the functions called from this function.
%self	Percentage of time spent in this function excluding the time spent in subroutines called from this function.
name	Function name.

Note: 100% represents the program execution time.

Local vs. remote debug

You can configure Genero Studio to debug a local or remote application.

Check your [active configuration](#) before beginning a debug session.

- Use a **local** Genero configuration to work on your local application.
- Use a **remote** Genero configuration or your **local** Genero configuration to access and debug your remote application.

Use these steps to debug an application where the binaries are located on a remote server (production server) and the sources are located locally.

1. Set a remote configuration.
2. Confirm that all binaries are on the remote server.
3. Create an empty **application node** with the same name as the 42r binary located on the remote server.
4. Set all required environment variables on the **application node**.

FGLLDPATH	Set to the path of the 42m and 42r files located on the remote server.
FGLRESOURCEPATH	Set to the path of the 42f files (and other resource files).
FGLSOURCEPATH	Set to the path of the source files located locally.

Reference

Reference information for the Debugger.

- [Debug context menu](#) on page 511
- [Views](#) on page 512
- [Supported debug commands](#) on page 516

Debug context menu

Commands on the **Debug** menu execute the application in debug mode.

Table 151: Debug Menu

Menu Option	Usage
Debug	Begins program execution, debugging up to the first breakpoint or interruption.
Execute with Profiler	Execute selected application with Profiler. See Profiler on page 510
Next	Executes the current line, and stops at the next source line in the <i>current</i> function. If this source code line is a call to another function, you can Step in or Step out.
Step in	Step into the current function, executing the next source line inside the <i>called</i> function.
Step out	Step out of the called function before it ends, returning to the next source line following the CALL statement.

Menu Option	Usage
	<p>Note: Some complex instructions such as CONSTRUCT or FOR loops need two steps before going to the next instruction: one step to prepare the statement, one step to execute it.</p>
Continue	Resumes program execution, until another breakpoint is reached or the program terminates.
Interrupt	<p>Interrupts the program execution, displaying the current line being executed and updates the Debugger views such as the Data view.</p> <p>Stops the execution of the program being debugged, returns control to the debugger, displays the current line being executed and updates the debugger views such as the Data view. This allows you to interrupt a program that is in an endless loop, or that is displaying a form, for example.</p>
Up	Displays the calling function. The up command advances toward the outermost frame each time it is processed. The Data view on page 513 is based on the new context.
Down	The opposite of Up: it comes back to the inner frame down to the last executed line.
Record Macro	Start recording Debugger commands.
Stop Record	Stop command recording and store the commands in a file.
Play Macro	Execute the recorded commands from the file.
Add/Delete Breakpoint	Sets / removes a breakpoint from the current line.
Enable/Disable Breakpoint	If checked, the current breakpoint is active; otherwise it is disabled, but still exists and can be reactivated later.
Abort Last Task	Terminates the debugger session.

Views

Information about Debugger views.

- [Command view](#) on page 512
- [Data view](#) on page 513
- [Watchpoints view](#) on page 514
- [Breakpoints view](#) on page 514
- [Backtrace view](#) on page 515

Command view

The Command view displays all of the debugger commands that have been executed. `fgldb` commands may be entered manually into the Command view.

The output of the command, if any, appears immediately below the command.

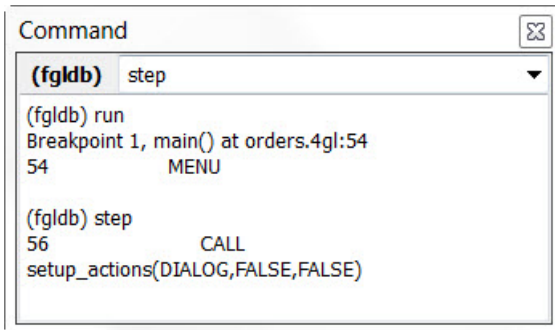


Figure 285: Command view

Data view

In the Data view you can examine or set the values of global, module, and function variables while your application is running.

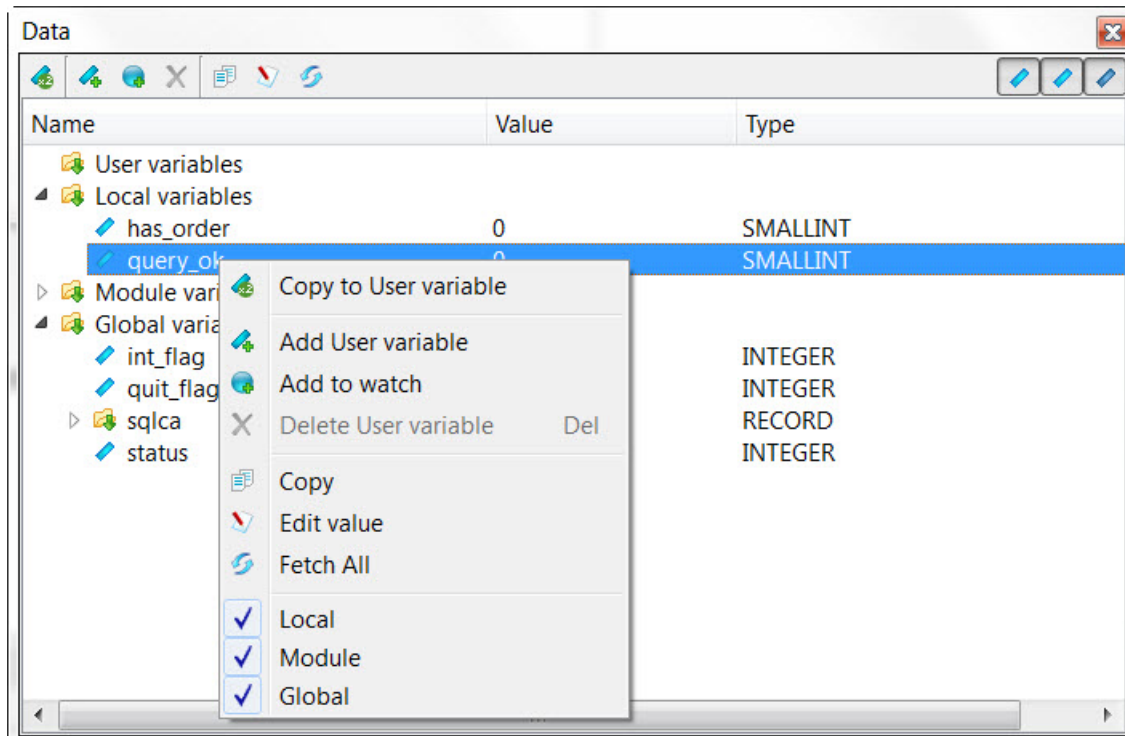


Figure 286: Data View

In the Data view, variables are organized into groups and displayed in a tree. Expand the group to display its variables and see each name, value, and type (record, column, data type). Double-click the value to edit variable values for testing purposes.

Use the integrated Toolbar to:

- **Copy** a variable to the User variables group
- **Add, Delete, or Duplicate** a User variable
- **Add** a variable to watch list
- **Display/Edit** a variable value
- **Show** Global, Module, and/or Local variables
- **Fetch All** variables

Right-click a variable to display a context menu with some of the same commands.

Watchpoints view

Each time the value of a watched variable changes, the program will stop, and the variable values will be displayed in the Watchpoints view.

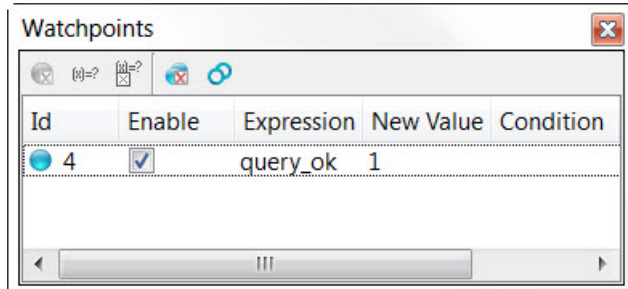


Figure 287: Watchpoints View

ID

The identification number of the watchpoint.

Enable

Check to enable the watchpoint, uncheck to disable to watchpoint.

Expression

The variable name on which the watchpoint is set.

New Value

The new value of the watchpoint variable.

Condition

A Boolean expression can specify that the [watchpoint is conditional](#); each time the value of the variable changes and the condition is TRUE, the program will stop.

The integrated Toolbar includes options to manage watchpoints.

Delete watchpoint

Delete the selected watchpoint.

Edit watchpoint condition

Edit the condition associated to the watchpoint.

Delete watchpoint condition

Removes the condition associated to the watchpoint.

Delete all watchpoints

Removes all watchpoints.

Disable all watchpoints

Disables all watchpoints.

Breakpoints view

This Breakpoints view displays information on all breakpoints that have been set.

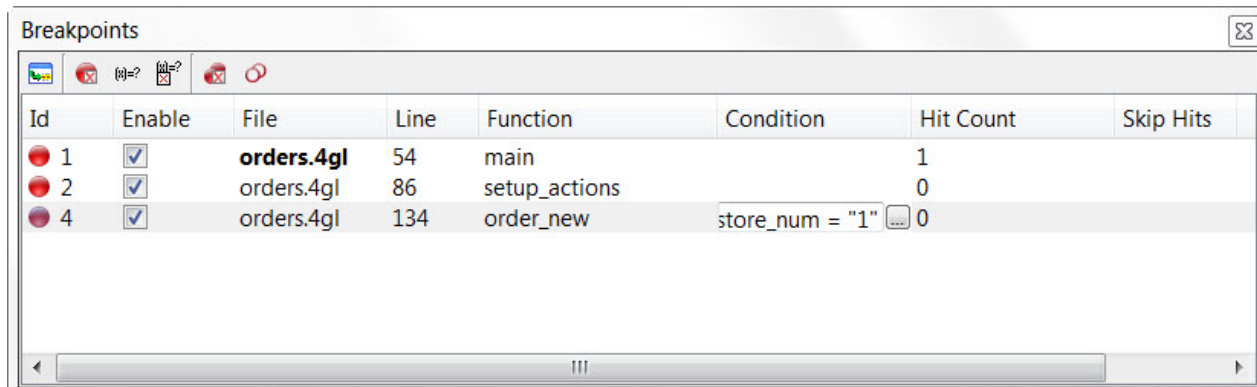


Figure 288: Breakpoints view

Id	Displays the number assigned to the breakpoint by the Debugger.
File, Line, Function	Identify the breakpoint's location in the source file.
Enable	Check to enable to breakpoint, uncheck to disable the breakpoint.
Condition	Specify conditions for the breakpoint. Create or edit a condition by clicking in the row's Condition column. If a condition is defined, it will be checked each time the line is to be executed. Execution will stop at this breakpoint only if the condition is true.
Hit Count	Total number of times a breakpoint has been hit
Skip Hits	Specifies the number of times the breakpoint should be ignored

The integrated Toolbar includes options to manage breakpoints. These options are also available by right-clicking on a line in the Breakpoints view.

Go to source code	Open the file containing the breakpoint. Double-clicking a line will also open the file containing the breakpoint.
Delete breakpoint	Delete the selected breakpoint.
Edit breakpoint condition	Edit the condition associated to the breakpoint.
Delete breakpoint condition	Removes the condition associated to the breakpoint.
Delete all breakpoints	Removes all breakpoints.
Disable all breakpoints	Disables all breakpoints.

Backtrace view

In the Backtrace view you can view or trace the functions called while running the program.

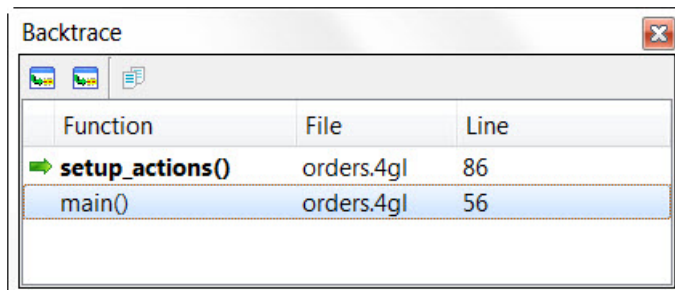


Figure 289: Backtrace view

The first line displayed is the current line. Then the calling function is displayed, and so on, up to the main function. The Backtrace view is display-only. Lines are added in the output while the program is running.

An arrow points to the currently executing function.

The integrated Toolbar includes options to navigate a backtrace.

Go to selected frame	Make the selected frame the current frame. The Data view will be updated with the variables available in the selected frame.
-----------------------------	--

Go to source code

Display the source code corresponding to the selected frame.

Supported debug commands

The Debugger supports a set of command-line commands.

break

Sets a breakpoint at the specified line or function

Syntax

```
b[reak] { line | module:line | function }
```

Note:

1. *line* is a source code line in the current module.
2. *module:line* is a source code line in a specific module.
3. *function* is a function name.

Usage

The `break` command sets a breakpoint at a given position in the program.

When the program is running, the debugger stops automatically at breakpoints defined by this command.

Example

```
(fgldb) break mymodule:5  
Breakpoint 2 at 0x00000000: file mymodule.4gl, line 5.
```

backtrace

Prints a summary of how your program reached the current state.

Syntax

```
backtrace  
bt
```

Usage

The `backtrace` command prints a summary of your program's entire stack, one line per frame. Each line in the output shows the frame number and function name.

Example

```
(fgldb) bt  
#1 addcount() at mymodule.4gl:6  
#2 main() at mymodule.4gl:2  
(fgldb)
```

clear

Clear breakpoint at the specified line or function.

Syntax

```
clear [function|line]
```

Note:

1. *function* - deletes any breakpoints set at entry to the specified function.
2. *line* - deletes any breakpoints set at or within the code of the line specified by number.

Usage

With the `clear` command you can delete specific breakpoints in your program. Use the `clear` command with no arguments to delete any breakpoints at the next instruction to be executed in the selected stack frame.

See the [delete](#) command to delete individual breakpoints by specifying their breakpoint numbers.

Example

```
(fgldb) clear mymodule:5
Deleted breakpoint 2
(fgldb)
```

continue

Continues the execution of the program after a breakpoint.

Syntax

```
c[continue] [ignore-count]
```

Note:

1. *ignore-count* defines the number of times to ignore a breakpoint at this location.

Usage

The `continue` command continues the execution of the program until the program completes normally, another breakpoint is reached, or a signal is received.

Example

```
(fgldb) continue
<..program output..>
Program exited normally.
```

define

Specifies a user-defined sequence of commands.

Syntax

```
define command-name { command1 command2 ... }
end
```

Note:

1. *command-name* is the name assigned to the command sequence.
2. *command* is a valid debugger command.
3. **end** indicates the end of the command sequence.

Usage

The `define` command allows you to create a user-defined command by assigning a command name to a sequence of debugger commands that you specify. You may then execute the command that you defined by entering the command name at the debugger prompt.

User commands may accept up to ten arguments separated by whitespace.

Example

```
(fgldb) define myinfo
> info breakpoints
> info program
> end
(fgldb)
```

delete

Removes breakpoints specified in a debugger session.

Syntax

```
del[ete] breakpoint
```

Note:

1. *breakpoint* is the number assigned to the breakpoint by the debugger.

Usage

The `delete` command allows you to remove breakpoints when they are no longer needed in your debugger session.

If you prefer you may disable the breakpoint instead. See the `disable` command.

Example

```
(fgldb) delete 1
(fgldb) run
Program exited normally.
(fgldb)
```

disable

Disables the specified breakpoint.

Syntax

```
disable breakpoint
```

Note:

1. *breakpoint* is the number assigned to the breakpoint by the debugger.

Usage

The `disable` command instructs the debugger to ignore the specified breakpoint when running the program.

Use the `enable` command to reactivate the breakpoint for the current debugger session.

Example

```
(fgldb) disable 1
(fgldb) run
Program exited normally.
(fgldb)
```

display

Displays the specified expression's value each time the program stops.

Syntax

```
disp[lay] expression
```

Note:

1. *expression* is your program's expression that you wish to examine.

Usage

The `display` command allows you to add an expression to an automatic display list. The values of the expressions in the list are printed each time your program stops. Each expression in the list is assigned a number to identify it.

This is useful in tracking how the values of expressions change during the program's execution.

Example

```
(fgldb) display a
1: a = 6
(fgldb) display i
2: i = 1
(fgldb) step
2: i = 1
1: a = 6
16      for i = 1 to 10
(fgldb) step
2: i = 2
1: a = 6
17      let a = a+1
(fgldb)
```

down

Selects and prints the function called by the current function, or the function specified by the frame number in the call stack.

Syntax

```
do[wn] [num]
```

Note:

1. *num* is the number of frames to move down the stack. The default is 1.

Usage

This command moves down the call stack, to the specified frame, and prints the function identified with that frame. To print the function called by the current function, use the `down` command without an argument.

```
(fgldb) down
#0 query_cust() at custquery.4gl:22
22 CALL cleanup()
(fgldb)
```

enable

Enables breakpoints that have previously been disabled.

Syntax

```
enable breakpoint
```

Note:

1. *breakpoint* is the number assigned to the breakpoint by the debugger.

Usage

The `enable` command allows you to reactivate a breakpoint in the current debugger session. The breakpoint must have been disabled using the `disable` command.

Example

```
(fgldb) disable 1
(fgldb) run
Program exited normally.
(fgldb) enable 1
(fgldb) run
Breakpoint 1, at mymodule.4gl:5
```

file

Specifies the name of the program being debugged.

Syntax

```
file filename
```

Note:

1. *filename* is the name of the program being debugged.

Usage

The `file` command can be used to change to a different file during a debugging session.

finish

Instructs the program to continue running until just after the function in the selected stack frame returns, and then stop.

Syntax

```
finish
```

Usage

The `finish` command instructs the program to continue running until just after the function in the selected stack frame returns, and then stop. The returned value, if any, is printed.

Example

```
(fgldb) finish
```

help

Provides information about debugger commands.

Syntax

```
h[elp] [command]
```

Note:

1. *command* is the name of the debugger command for which you wish information.

Usage

The `help` command displays a short explanation of a specified command.

Enter the `help` command with no arguments to display a list of debugger commands.

Example

```
(fgldb) help delete
```

info

Describes the current state of the program.

Syntax

```
i[info] [ { breakpoints | sources | program | stack | files |
           line (linespec | module:line | function) } ]
```

Note:

1. **breakpoints** lists the breakpoints that you have set.
2. **sources** prints the names of all the source files in your program.
3. **program** displays the status of your program.
4. **stack** summarizes how your program got where it is.
5. **files** lists the names of the executable file and core dump files currently in use, and the files from which symbols were loaded.
6. **line** *linespec* maps the specified source code line to program addresses.
7. **line** *module:line* maps the specified source code line to program addresses.

8. **line** *function* prints the program addresses for the first line of the function named *function*.

Usage

The `info` command describes the state of your program.

The command `info line linespec` prints the starting and ending addresses of the compiled code for the source line specified. See the `list` command for all the ways that you can specify the source code line.

Example

```
(fgldb) info sources
Source files for which symbols have been read in:
```

```
mymodule.4gl, fglwinexec.4gl, fglutil.4gl, fgldialog.4gl, fgldummy4js.4gl
(fgldb)
```

list

Prints source code lines of the program being executed.

Syntax

```
l[ist]
```

Usage

The `list` command prints source code lines of your program, including the current line.

Example

```
(fgldb) run
Breakpoint 1, at mymodule.4gl:5
5   call addlist()
(fgldb) list
5   call addlist()
6   call addname()
.
14  end function
(fgldb)
```

next

Continues running the program by executing the next source line in the current stack frame, and then stops.

Syntax

```
n[ext]
```

Usage

The `next` command allows you to execute your program one line of source code at a time. The `next` command is similar to `step`, but function calls that appear within the line of code are executed without stopping. When the next line of code at the original stack level that was executing when you gave the `next` command is reached, execution stops.

After reaching a breakpoint, the `next` command can be used to examine a troublesome section of code more closely.

Example

```
(fgldb) next
5 call addlist()
(fgldb) next
6 call addname()
(fgldb)
```

output

Prints only the value of the specified expression, suppressing any other output.

Syntax

```
output expression
```

Note:

1. *expression* is your program's expression that you wish to examine.

Usage

The `output` command prints the current value of the expression and nothing else, no newline character, no "expr=", etc.

The usual output from the debugger is suppressed, allowing you to print only the value.

Example

```
(fgldb) output b
```

print

Displays the current value of the specified expression.

Syntax

```
p[rint] expression
```

Note:

1. *expression* is your program's expression that you wish to examine.

Usage

The `print` command allows you to examine the data in your program.

It evaluates and prints the value of the specified expression from your program, in a format appropriate to its data type.

Example

```
(fgldb) print b
$1=5
(fgldb)
```

quit

Terminates the debugger session.

Syntax

```
q[uit]
```

Usage

The `quit` command allows you to exit the debugger.

Example

```
(fgldb) quit
<system prompt>
```

run

Starts the program executing until a breakpoint is reached or the program terminates normally.

Syntax

```
r[un] [arg1 arg2 ... ]
```

Note:

1. *arg* is an argument to be passed to your program.

Example

```
(fgldb) run
Breakpoint 1, at mymodule.4gl:3
3   call addcount()
(fgldb)
```

set

Allows you to set the value of specific variables for the duration of the debugger session.

Syntax

```
set { | _environment varname[=value] | verbose {on | off} | annotate }
```

Note:

1. *varname* is the environment variable to be set to *value*.

Usage

The `set` command changes the values of the these variables:

When setting an **environment** variable, *value* may be any string. If the *value* parameter is omitted, the variable is set to a null value. The variable is set for your program, not for the debugger itself.

When **verbose** is set to **on**, the debugger will display additional messages about its operations, allowing you to observe that it is still working during lengthy internal operations.

Example

```
set verbose on
```

Important: On UNIX™ systems, if your SHELL variable names a shell that runs an initialization file, any variables you set in that file affect your program. You may wish to move setting of environment variables to files that are only run when you sign on, such as `.login` or `.profile`.

source

Executes a file of debugger commands.

Syntax

```
source commandfile
```

Note:

1. *commandfile* is the name of the file containing the debugger commands.

Usage

The source command allows you to execute a command file of lines that are debugger commands. The lines in the file are executed sequentially.

The commands are not printed as they are executed, and any messages are not displayed. Commands are executed without asking for confirmation.

An error in any command terminates execution of the command file.

Example

```
(fgldb) source mycommands
```

signal

Sends an INTERRUPT or QUIT signal to the program.

Syntax

```
sig[nal] SIGINT
```

```
sig[nal] SIGQUIT
```

Usage

The `signal SIGINT` command resumes execution of your program where it has stopped but immediately sends an INTERRUPT signal. The source line that was current when the signal was received is displayed.

Use `signal SIGQUIT` to send a QUIT signal.

Example

```
(fgldb) signal SIGINT
Program exited normally.
16      for i = 1 to 10
(fgldb)
```

step

Continues running the program by executing the next line of source code, and then stops.

Syntax

```
s[tepl] [count]
```

Note:

1. *count* defines the number of lines to execute before stopping.

Usage

The `step` command allows you to "step" through your program executing one line of source code at a time. When a function call appears within the line of code, that function is also stepped through.

A common technique is to set a breakpoint prior to the section or function that is causing problems, run the program until it reaches the breakpoint, and then step through it line by line.

Example

```
(fgldb) step
4 call addlist(a)
(fgldb)
```

Important: The `step` command cannot be used to step through a function that was compiled without debugging information. Execution continues until it reaches a function that does have debugging information.

until

Continues running the program until the specified location is reached.

Syntax

```
u[ntil] { line | module:line | function }
```

Note:

1. *line* is a source code line in the current module.
2. *module:line* is a source code line in a specific module.
3. *function* is a function name.

Usage

The `until` command continues running your program until either the specified location is reached, or the current stack frame returns. This can be used to avoid stepping through a loop more than once.

Example

```
(fgldb) until addcount()
```

up

Selects and prints the function that called this one.

Syntax

```
up [num]
```

Note:

1. *num* is the number of lines to move up the stack. The default is 1.

Usage

The `up` command advances toward the outermost frame, to frames that have existed longer.

watch

Sets a watch point for an expression which stops execution of your program whenever the value of the expression changes.

Syntax

```
watch expression [boolean-expression]
```

Note:

1. *expression* is the expression to watch.
2. *boolean-expression* is an optional boolean expression.

Usage

The watch point stops the program execution when the value of the expression changes.

If *boolean-expression* is provided, the watch point stops the execution of the program if the expression value has changed and the *boolean-expression* evaluates to TRUE.

Example

```
(fgldb) watch i if i >= 3
```

where

Alias for `backtrace` command.

Syntax

```
w[here]
```

Usage

The `where` command, like `backtrace`, prints a summary of your program's entire stack, one line per frame. Each line in the output shows the frame number and function name.

Example

```
(fgldb) where
#1 addcount() at mymodule.4gl:6
#2 main() at mymodule.4gl:2
(fgldb)
```

Unsupported commands

These debugger commands are not supported in Genero Studio:

- `define`
- `source`
- `tbreak`

- tty

Source Code Management - SVN

Genero Source Code Management (SCM) enables collaborative sharing and maintaining of the files in Genero projects.

- [What is Genero Source Code Management?](#) on page 529
- [SCM Usage](#) on page 530
- [SCM Reference](#) on page 540

What is Genero Source Code Management?

Genero Source Code Management (SCM) enables collaborative sharing and maintaining of the files in Genero projects.

A Subversion client must be installed on your local machine. Genero Studio for Windows™ includes Apache's Subversion client. Genero Studio for GNU/Linux relies on Subversion 1.6.2 or later, which must have been installed on the system. For more information about Apache's Subversion, see: <http://subversion.apache.org/>

Options in SCM allow you to manage the files in the Subversion repository. The Subversion repository stores current and historical versions of the files, allowing you to recover older versions of your data when necessary. Your files (called working copies) are stored in an ordinary directory on your local file system. When you make changes to these working copies, you commit the changed files to the SVN repository. If projects or project files have already been committed to a repository, you can load copies into a working directory on your local system.

A SVN repository often holds the files (or source code) for several projects; usually, each project is a subdirectory in the repository's file system. Your working copy will usually correspond to a particular subtree of the repository.

If a Project is under Source Code Management, the icons for the nodes will have a status icon superimposed when you open the Project.

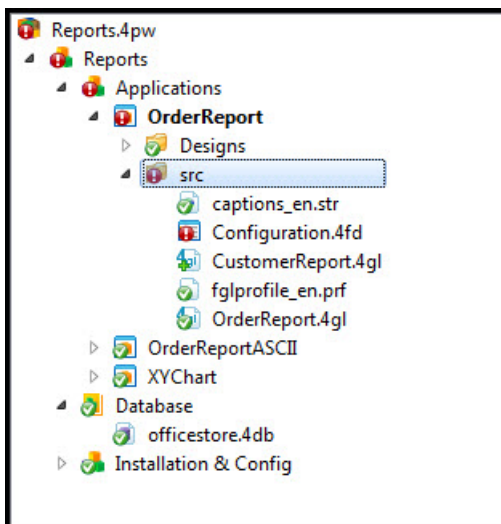






Figure 290: Project view with status icons

Table 152: Status icons and their meanings

Icon	Description
	Item is unchanged since it was last committed to the repository.
	Item has been changed and has not been committed yet. The application and the project will also have change indicators.
	Item is scheduled for addition to the repository.
	Item is in conflict with updates received from the repository.

SCM Usage

Information about using Source Code Management.

- [Checkout files](#) on page 530
- [Add files](#) on page 531
- [Commit / Review changes](#) on page 531
- [Locking](#) on page 532
- [Revert changes / Un-add files](#) on page 534
- [Delete files](#) on page 534
- [Update / Update All](#) on page 534
- [Cleanup](#) on page 535
- [Copy working files and directories](#) on page 535
- [Revert from a single revision](#) on page 535
- [Merge and revert](#) on page 535
- [Move a working copy \(Switch\)](#) on page 536
- [Create patch](#) on page 536
- [Apply patch](#) on page 536
- [Browse repository](#) on page 537
- [View log information](#) on page 537
- [Specify the revision range for logs](#) on page 537
- [Blame](#) on page 537
- [Diff with revised file](#) on page 539

Checkout files

You can checkout a project from the SCM menu or directly from the location of your new working directory.

Before you begin, you must have access to an SVN Repository that has been set up to store your versioned files. Contact your system administrator or Subversion vendor for information and documentation on the set up of Subversion and repositories.

1. Identify the directory or directories on your local system that will contain the files you checkout from the repository.
2. Check out the files from the repository to your checkout directory.

Option

Description

Select SCM >> Checkout from the main menu.

An [SVN Checkout dialog](#) will guide you through the checkout steps.

Navigate to the directory that will serve as your new working directory. If the directory does not exist, create it. Right-click on the directory and select SCM >> Checkout.

An [SVN Checkout dialog](#) will guide you through the checkout steps and the checkout directory path will be filled in for you.

Once the files have been added to your checkout directory, you are prompted to select a project file (4pw). If one does not exist, you can create a new project and save it in the checkout directory.

The files to be versioned must be stored in the checkout directory.

3. Use **File >> New** to create files and store them in the checkout directory and the project structure as needed.
4. Any files from the checkout directory that have been added to the Project can be [committed](#) to the repository.

At this point, you can use the SCM menu commands to handle versioning for your files in the checkout directory.

Note: Although the 4pw project file is used internally by Genero Studio, we recommend that you include it in the checkout directory and commit it to the repository to share it with other developers. Although the 4pw and .deps files are in the checkout directory, they do not have to be added to the Project Structure.

Add files

Complete this procedure to add unversioned files found in the checkout directory.

1. Select **Window >> Views >> SVN Status**.
2. From the SVN Status view, navigate to your checkout directory if it is not already displayed.
3. Click **Add mode**.

All unversioned files found are listed. The files/folders with 'not a working copy' text status or none property status within an unversioned folder will also be shown.

4. Check those files you wish to add to the repository.
5. Click the **Add** button.

The files are added, their text status changes to 'added'. They are not yet committed.

6. To commit these newly added files, select Add Mode again to change views and select [Commit](#).
7. To revert this action (for files that have been added but not committed), see [Revert changes](#).

Commit / Review changes

To commit your updated version of a file to the repository, right-click on the file in your Project and choose **SCM >> Review Changes**.

From the SVN Status view, check the box in the **Select** column for the files you wish to commit, and press the **Commit** button at the top of the view. This will invoke the **SVN Commit** dialog to commit any checked file.

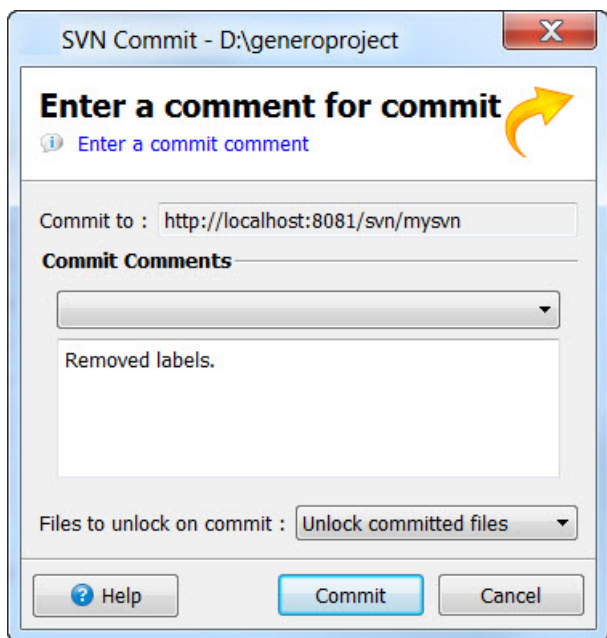


Figure 291: SVN Commit dialog

Commit Comments

Enter a comment in the text box, or select a comment from the list of recent comments.

Files to unlock on commit

Unlock committed files (default)

Unlock only those locked files that are committed. Any locked file that is not committed will not be unlocked on commit.

Unlock all files

Unlock all locked files after commit. If some of the locked files are modified but not committed, you are warned that some of the modified files will be unlocked and asked whether to proceed with commit.

Keep all locks

No file is unlocked after commit.

Locking

Locking a file provides exclusive rights to a user for changing that file in the repository. As long as the file in the repository is locked by a user, no other user can change (commit) that file in the repository. This helps to avoid conflicting commits.

Subversion provides a property `svn:needs-lock`. This property can be set on files only. The `svn:needs-lock` property sets the file to read-only when checked out. A user who has checked out the file will have to lock the file to modify it in the working copy. If another user wants to modify the same file, they will have to first acquire the lock on the same file and is informed that the file is already locked by another user. In this way a locked file cannot be modified simultaneously which avoids possible conflicts.

SVN Lock strategies

There are three user strategies for working with locks in Genero Studio; use `svn:needs-lock` on some files, on all files, or not at all.

Use `svn:needs-lock` on some files

There are some file types like binary files, which can not be merged. Setting `svn:needs-lock` on those files specifies that only one user can modify them at one time. You must lock the file before modifying or check for existing locks. If the file is not locked, you may have to merge the changes on commit. Locked files must be unlocked after commit.

Use `svn:needs-lock` on all files

With this strategy, no commit conflicts can occur because no two users can modify the same file at the same time. You will have to lock the file before modifying it. Locked files must be unlocked after commit.

Not using the `svn:needs-lock` property

This strategy can be used when all the files can be merged manually in case of conflicts. In case of binary files it is not possible to merge the changes, so modifying the files without locking must be avoided. If the file is not locked and modified, you may have to manually merge the changes in case of commit conflicts.

Set `svn:needs-lock` property

Setting `svn:needs-lock` on a file specifies that only one user can modify it at one time.

1. Use **SCM >> Properties** from the right-click context menu on a file to make changes to SVN properties.
2. Use the plus sign to add a new property. Select `svn:needs-lock` from the **Property name** list.
3. Set the **Property value** to `*`.

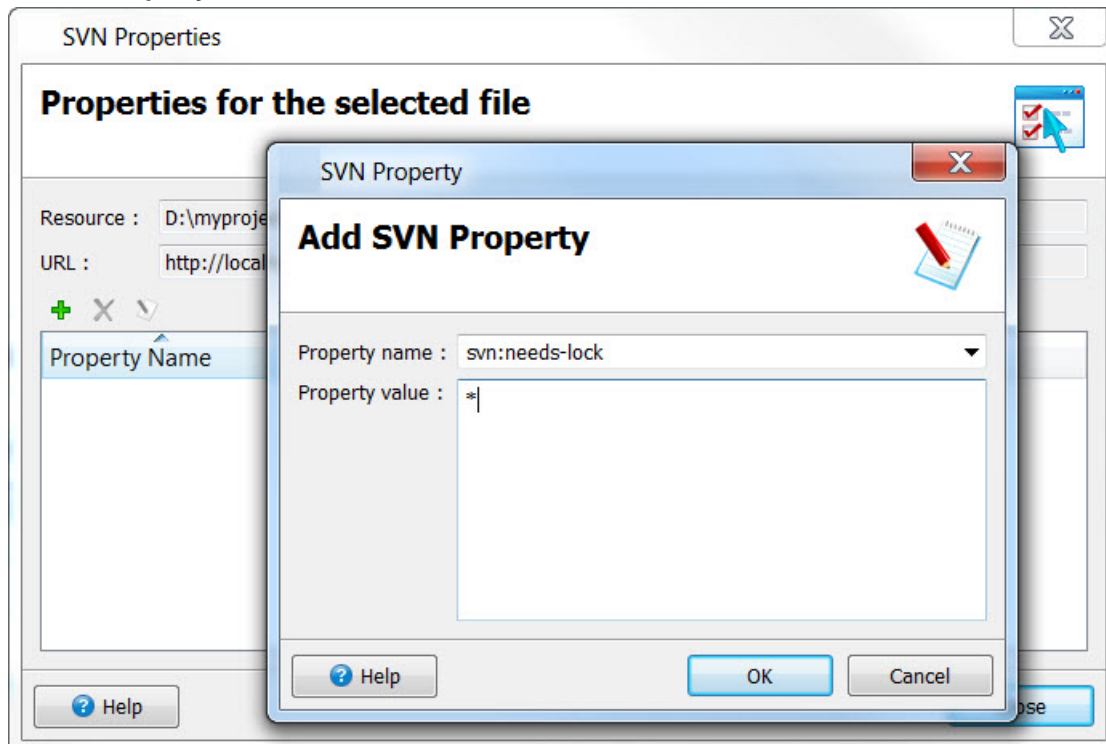


Figure 292: Set `svn:needs-lock`

Lock a file

If `svn:needs-lock` has been set on a file, the current user can lock the file.

1. From the project or **SVN Status** view, right-click on the file and select **Lock**.

2. Enter a comment and press the **Lock** button.

Revert changes / Un-add files

You can discard changes to files not yet committed, and you can unadd files which have been added but not yet committed.

1. Open the SVN Status view.
If you are in **Commit mode**, the Revert button displays. If you are in **Add mode**, the Unadd button displays.
2. Check those files you wish to revert or unadd.
3. Click **Revert** (Commit mode) or **Unadd** (Add mode).
4. A dialog asks you to confirm your selection.
Click **Yes**.

For modified files, the change is reverted and the previously committed version of the files is restored to your local (working) copy. For added files, the file is changed from 'added' to 'unversioned'.

Delete files

The **Delete** command deletes a file from your checkout directory and from the repository.

1. Right-click the file.
2. Select **Delete** from the context menu.
Your local copy is deleted. The next time you commit, the Projects view will request the source code management system delete the file from its repository.

Update / Update All

Update / Update All updates any outdated files in your project with the latest version stored in the repository.

The Update options are presented as buttons in the Status view Toolbar, or can be accessed right-clicking on a specific file selected from the project tree in the Projects view.

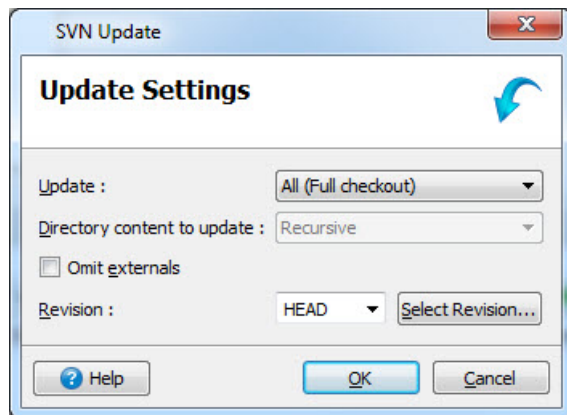


Figure 293: SVN Update dialog

Specify what files to update in the SVN Update dialog.

In the **Update** combobox, specify which files to update. Choices are:

- **All** (a full checkout)
- **Selection** (selected files)

In the **Directory content to update** combobox, specify the depth for the update.

- With an All update, the depth is automatically set to Recursive.

- With a Selection update, the choices are Recursive, None, Immediate files only, or Immediate files and folders only. For updates from the Projects view using the context menu, the depth will always be None.

Select the **Omit externals** checkbox to exclude external items. External items can be of two types:

- If a working copy contains items from some other location of same repository, these items are called external for this working copy.
- If there are two repositories and one of the repositories (R1 for example) contains items from other repository (R2 for example), these items are called "externals" for repository R1.

In the **Revision** combobox, select the revision to be updated. The **Select Revision** button opens the **SVN Log Select revision** dialog.

Cleanup

The **Cleanup** command cleans up the working copy, removing stale locks.

1. In the Projects or Files view, right-click on a file or directory.
2. Select **SCM >> Cleanup**.

Copy working files and directories

A Subversion working copy is an ordinary directory tree on your local system, containing a collection of files. You can create another copy of your project, project files, or directories, to allow parallel development, for example.

Before you begin, you must have access to an SVN repository that has been set up to store your versioned files. Contact your system administrator or Subversion vendor for information and documentation on the set up of Subversion and repositories.

1. Identify the working directory you wish to copy.
2. Copy the files.

Option	Description
Select SCM >> Copy from the main menu.	Complete the SVN copy dialog .
In the Files tab, navigate to the working directory you wish to copy. Right-click on the directory and select SCM >> Copy.	Complete the SVN copy dialog . The From section populated for you.

Revert from a single revision

Follow this procedure to revert changes from a single revision.

1. Select **SCM >> Merge/Revert...** from the main menu to open the [Merge/Revert dialog](#) on page 545.
2. Enter the repository URL and version, leaving the Range ends with a different URL check box not selected.
3. Specify the working copy directory to be updated in the working copy path.
4. Select the desired merge options
5. Click **Merge/Revert**.

Merge and revert

Follow this procedure to merge and revert changes from a range of revisions.

1. Select **SCM >> Merge/Revert...** from the main menu to open the [Merge/Revert dialog](#) on page 545.
2. Enter the repository URL and Revision.
3. Check the Range ends with a different URL check box and specify the revision URL and version to close the range.
4. Specify the working copy directory to be updated in the working copy path.

5. Select the desired merge options
6. Click **Merge/Revert**.

Move a working copy (Switch)

The Switch subcommand allows you to move a working copy to a new branch of the repository.

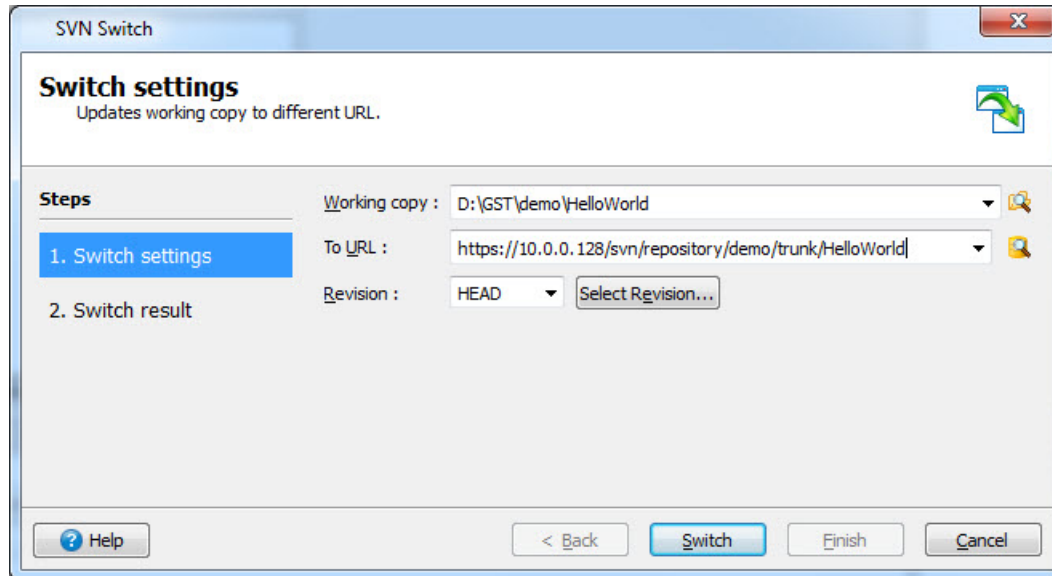


Figure 294: SVN Switch dialog

Create patch

You can create a patch file containing the differences between your working copies and the corresponding files in the repository. The created file is in *Unidiff* format.

If one of the checked files is unversioned or missing, the Create Patch command is disabled.

To create the patch file:

1. In the [SVN Status view](#) on page 543, check the file(s) that contain the modifications to be included in the patch.
2. Click **Create Patch**.
3. In the Save file dialog, select a directory and enter a name for the patch file.

The patch file is created and opened in Code Editor.

Apply patch

The **Apply patch** command applies a patch file to files in the repository.

This feature is available for Windows™ and Linux™ users only.

1. Identify the patch file to be used.
2. Select **SCM >> Apply Patch...**
3. Complete the [Apply Patch dialog](#) on page 544.
4. Press **Apply**.

A page containing the output of the command is displayed. The output will indicate Successful when all the files from the patch file are successfully patched. If only some of the files are patched successfully, the output will indicate Aborted.

Browse repository

The SVN Repository lists the files in your selected repository and branch.

1. Select **Window >> Views >> SVN Repository** to open the [SVN Repository Browser](#).
2. Enter the URL of the repository or select it from the drop down list.
3. Select the desired version. To select a specific revision within the version, click the [Select Revision icon](#) and modify the selection range.
4. Click the Refresh icon.

View log information

Complete this procedure to view information about the revisions for a specified repository.

1. Select **SCM >> Show Log**.
2. Enter the URL of the repository. Click the Browse repository icon to display the [Repository browser](#).
3. Select the desired version.
4. Click the Fetch the revisions in the revision range icon (single blue arrow).

The SVN Log view displays. The Revisions, Actions, Author, Date, and Messages are displayed in the log. The icons for Actions indicate the [status](#).

5. To limit the list of revisions displayed, open and complete the [Revision log dialog](#).
6. Select a specific revision to display information about that revision.
7. The comments and actions/files associated with that revision are displayed.

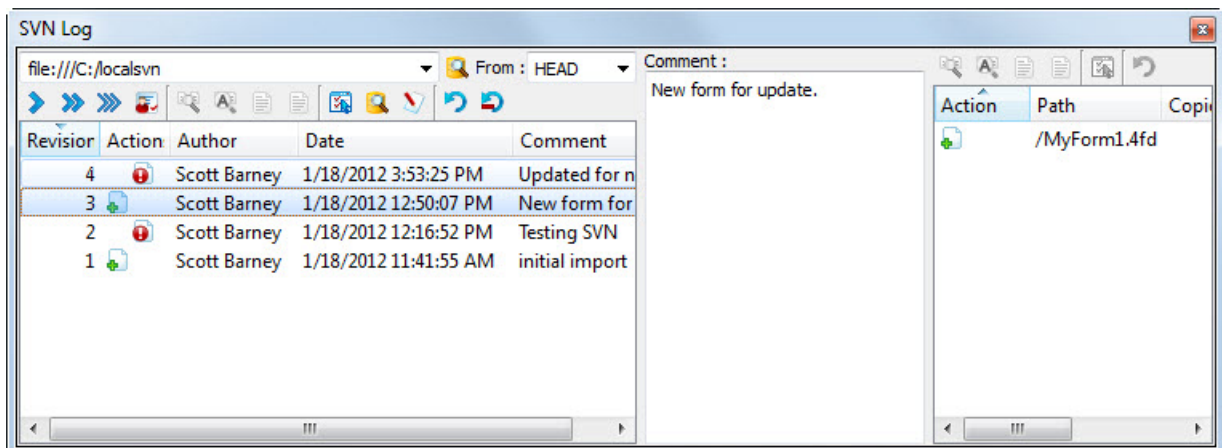


Figure 295: SVN Log view

Specify the revision range for logs

You can select the range of revision logs to display using the Revision log options dialog.

1. Click the **Open revision log options** icon.
2. Provide the range in the from: and to: fields, respectively.
3. Click **OK**.
The view is updated to display only those revisions in the range specified.
4. Use the **Fetch** actions to retrieve more log entries.

Blame

The **Blame** command shows author and revision information inline for specified files or URLs.

The Blame view displays the file as read-only in the Code Editor. Each line of text is annotated in the left margin with the author (username) and the revision number for the last change to that line. On clicking the left margin, the log comment for the revision displays.

Operations available in Blame view

Blame view operations can be accessed from the Toolbar.

Operation	Description
Show log	Shows the revision log view and the log entries starting from the selected revision to the first revision.
Blame in New Tab	Shows a new Blame view, where the maximum blame revision is the selected revision and the minimum blame revision is the minimum revision for the current Blame view.
Blame	Updates the current Blame view, where the maximum blame revision is the selected revision and the minimum blame revision is the minimum revision for the current blame view.
Diff with previous revision	Shows the difference between the selected blame revision and the previous revision.
Next / Previous	If you have viewed blame multiple times in the Blame view, you can scroll through the views using the next and previous options.

Access the Blame view

As a prerequisite:

The file must be a versioned file. The blame option is not available for Added files.

Access the Blame view for author and revision information.

1. Open one of these views:

- File Browser
- Projects
- SVN Status
- SVN Log (If the viewed log is about a file, then the blame option is available.)
- SVN Repository Browser

2. Select one or more files.

If multiple files are selected, separate blame views are opened for each file.

3. Right-click on the file and select a Blame option.

Option

SCM >> Blame

SCM >> Blame...

Description

Blame information fetches with the revision range from 1 to HEAD.

SVN Blame configuration dialog displays. If multiple files are selected, the dialog displays once and the selected settings are used for all files.

- Specify the range.
- Click **OK**.

If you are in the SVN Log view:

- If one or two revisions are selected, both **Blame** and **Blame...** are available.

- If one revision is selected, the **from** is 1 and the **to** is the revision number selected.
- If two revisions are selected, the **from** is the smaller revision number and the **to** is the greater revision number.
- If no revision is selected, the **from** is 1 and the **to** is HEAD.

If you are in the SVN Repository Browser view, the **to** is currently selected revision.

Diff with revised file

Ctrl-Click to select two revisions of a file in the log. Select **Diff** from the right-click Menu to call the [Using the Diff tool](#) on page 380 utility to compare the committed file with the file that you have revised.

Integrate bug tracking

Changes made in your project may also be related to a specific bug or issue ID. If you use a bug tracking system (such as Bugzilla) you can associate the changes you make in your Source Code Management system (such as Subversion) to its bug tracking ID.

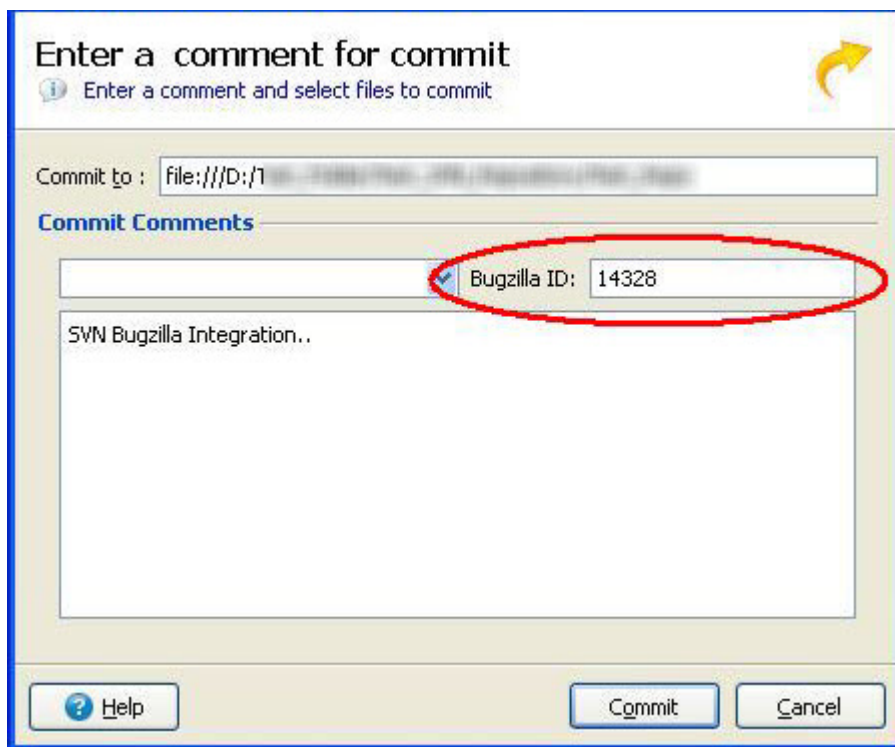


Figure 296: Source Code Management capturing bug ID

Revision	Actions	Author	Date	Message	Bugzilla ID
81839		tc	1/7/2010 11:02:49 PM	Sync w/ manuals CVS.	
81827		haya	1/7/2010 7:30:36 PM	Fixed Bug: Focus shifts to SVN Reposito...	14770
81826		nipr	1/7/2010 5:42:12 PM	Fixed Bug 16130 - Action 'Locate in Pr...	16130
81825		nipa	1/7/2010 4:54:59 PM	Updated the ui file names.Validator : nida	
81823		panm	1/7/2010 4:29:23 PM	Modification in SVN Bugzilla Integration....	

Figure 297: Source Code Management displaying bug ID

Integrate bug tracking

In Subversion, add specific 'bugtraq' properties on the repository. See [SVN documentation](#) for more information about usage for each of these properties.

bugtraq:logregex	Activates the bug tracking system.
bugtraq:message	Activates an additional field on the Commit dialog to prompt the user for an issue number.
bugtraq:label	Label for the additional field on the Commit dialog.
bugtraq:warnifnoissue	Display a Warning message if the added field is empty when doing a commit.
bugtraq:url	URL of the bug in the tracking system.

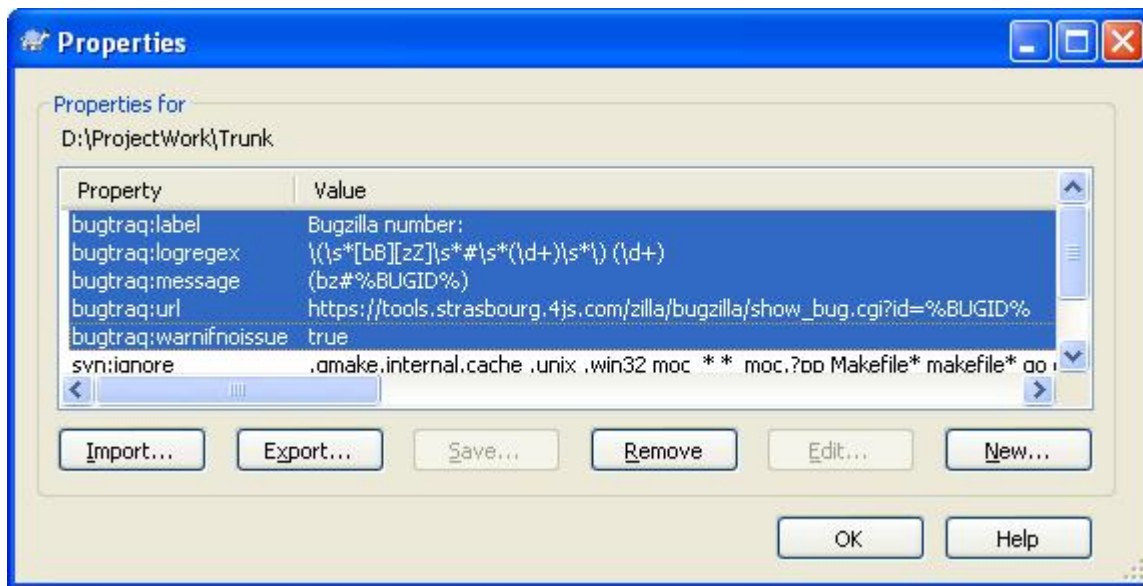


Figure 298: Add properties on SVN repository

SCM Reference

Reference information for Source Code Management.

- [Views](#) on page 540
- [Dialogs](#) on page 544
- [Specify a Subversion client](#) on page 549
- [SVN error messages](#) on page 549

Views

To display views that are not visible, use the **Window >> Views** menu.

The SVN Repository view is covered in the topic [Browse repository](#) on page 537.

- [SVN Log view](#) on page 541
- [SVN Locks view](#) on page 541
- [SVN Status view](#) on page 543

SVN Log view

The **SVN Log** view displays information about the revisions in a repository.

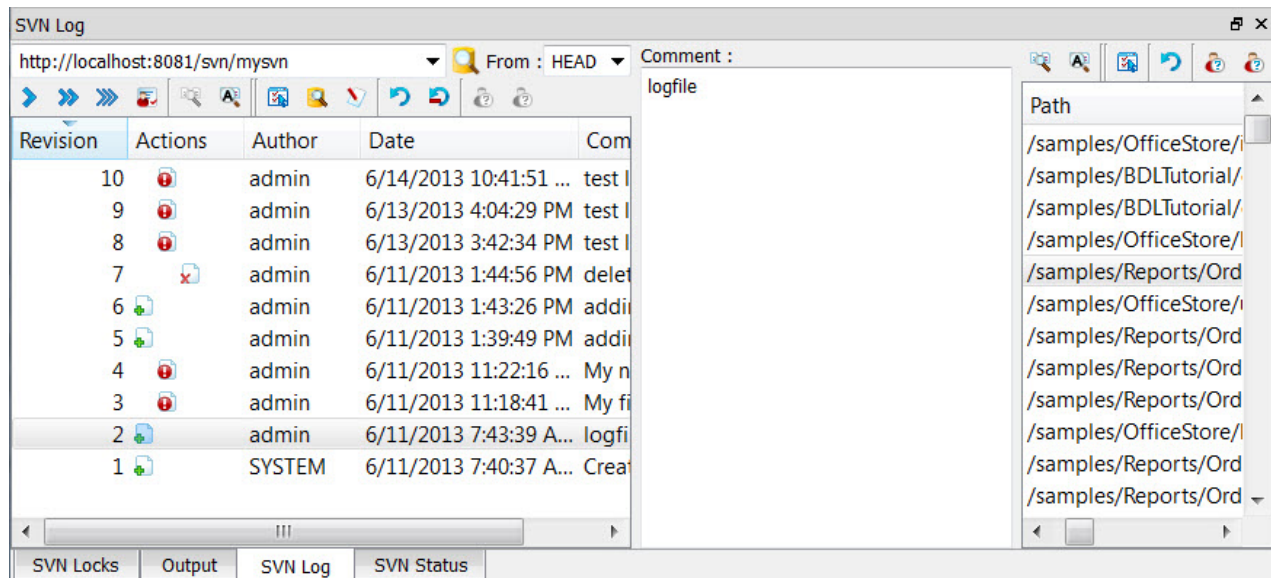


Figure 299: SVN Log view

Repository URL

Location of repository of which to view logs.

Repository Browser

Display Repository Browser. See [Browse repository](#) on page 537.

Fetch entries in the revision range

Fetch entries in current revision range.

Fetch next 100 entries

Fetch next 100 entries in current revision range.

Fetch next 5000 entries

Fetch next 5000 entries in current revision range.

Open revision log

Specify revision range. See [Specify the revision range for logs](#) on page 537

Open diff view

See [Diff with revised file](#) on page 539.

Open property diff view

See diff information for SVN properties.

Properties

See properties set on file. [SVN Properties dialog](#) on page 548.

Show repository browser

Display Repository Browser. See [Browse repository](#) on page 537.

Edit log comment

Edit log comment of selected entry.

Revert changes from selected revisions

See [Merge and revert](#) on page 535.

Revert to this revision

See [Merge and revert](#) on page 535.

Show Blame information

See [Blame](#) on page 537.

SVN Locks view

The SVN Locks view shows the lock and lock information for locked files in the given checkout directory.

From the SVN Locks view you can unlock your locked copies and steal locks from other users if forced acquisition of a lock is necessary.

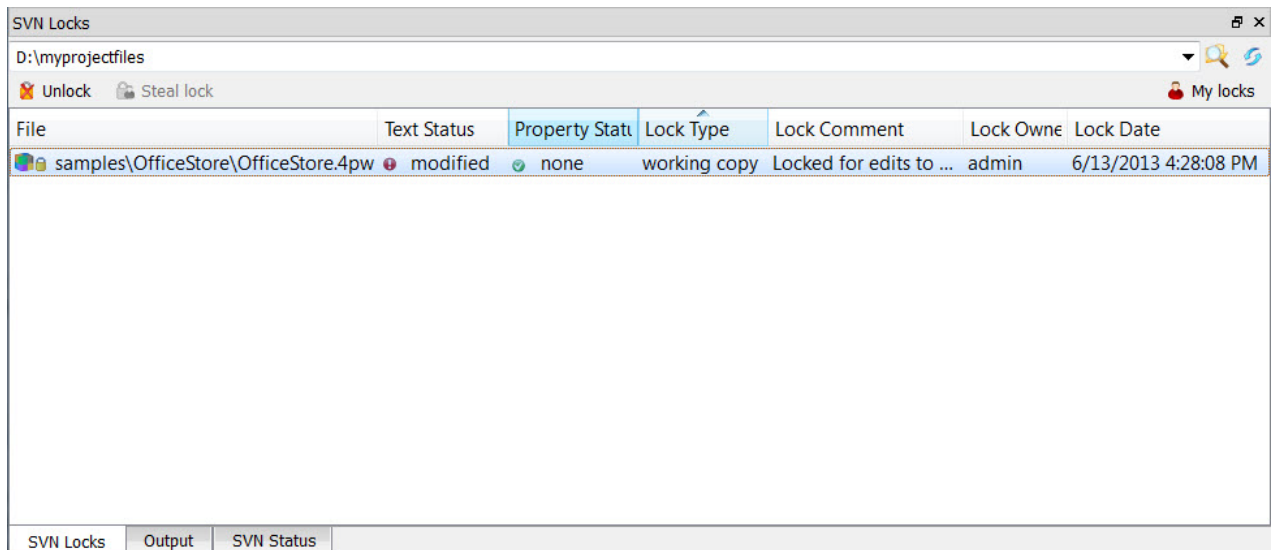


Figure 300: SVN Locks view

Checkout directory path

Enter a working copy directory path to view locked files in that directory.

Browse

Navigate to a checkout directory to populate the checkout directory path.

Refresh

Refresh the data in view.

Unlock

Unlock the selected file.

Steal Lock

Forcefully acquire the lock if locked by another user.

My locks

Display locks owned by the current user.

File list

- **File:** Displays file path relative to checkout directory path.
- **Text Status:** File text status in current working copy.
- **Property Status:** File property status in current working copy.
- **Lock Type:** Information about whether the file is locked in current working copy or not.
 - Working copy: The file is locked in current working copy.
 - Repository only: The file is locked in repository not in current working copy.
- **Lock Owner:** User who has locked the file.
- **Lock Comment:** Lock comment given by the lock owner when the file was locked.
- **Lock Date:** Date when the file was locked.

SVN Status view

The SVN Status view displays information about changes to your working copy.

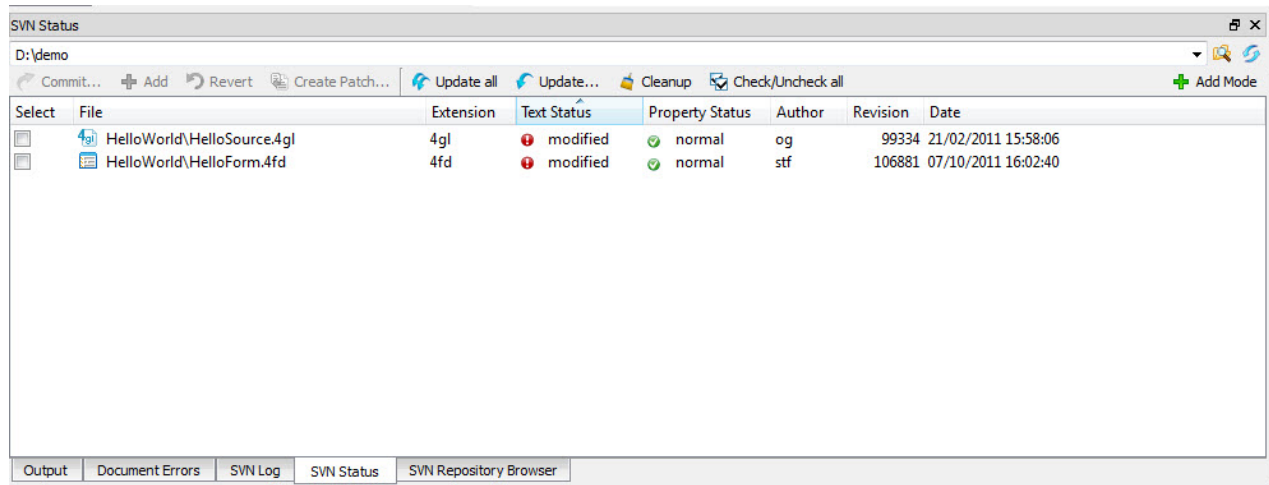


Figure 301: SVN Status view

Check the file or files to affect. The spacebar can be used to check / uncheck the current item. Toolbar icons allow you to execute commands on the selected files. The files displayed, and the icons enabled, depend on the mode and the status of the files selected.

The text status of a file can be:

added

The file has been added to the repository and needs only to be committed.

modified

The file has been modified in the checkout directory and the update has not yet been committed.

normal

A committed file in the repository.

unversioned

The file has not yet been added to the repository. After being added, it should be committed.

conflicted

The file was modified in both the repository and checkout directory, and Subversion cannot resolve the conflict. Use the menu option [Merge](#).

not working copy

All the files within an unversioned folder will have this status.

missing

The file has been deleted from the checkout directory. If you commit a file marked as missing, it will be skipped during commit and will remain as it is.

incomplete

If the checkout has stopped without completion, the directory will have this status.

SVN Status modes

Commit mode

Files that can be committed are displayed. Files that cannot be committed but are not supported in Genero Studio are displayed (with the exception of files with 'ignored' and 'external' status). Unversioned files are not displayed. This is the default mode.

Add mode

Files with a status of added, copied/moved, unversioned, or 'not a working copy' are displayed.

To switch between modes, click the **Add mode** button.

Dialogs

Information about SCM dialogs.

- [Apply Patch dialog](#) on page 544
- [Merge/Revert dialog](#) on page 545
- [The SVN Copy dialog](#) on page 546
- [SVN Checkout dialog](#) on page 546
- [SVN Lock dialog](#) on page 548
- [SVN Properties dialog](#) on page 548
- [SVN Repository view](#) on page 549

Apply Patch dialog

The Apply Patch dialog is used to apply a patch file. Only text modifications are applied.

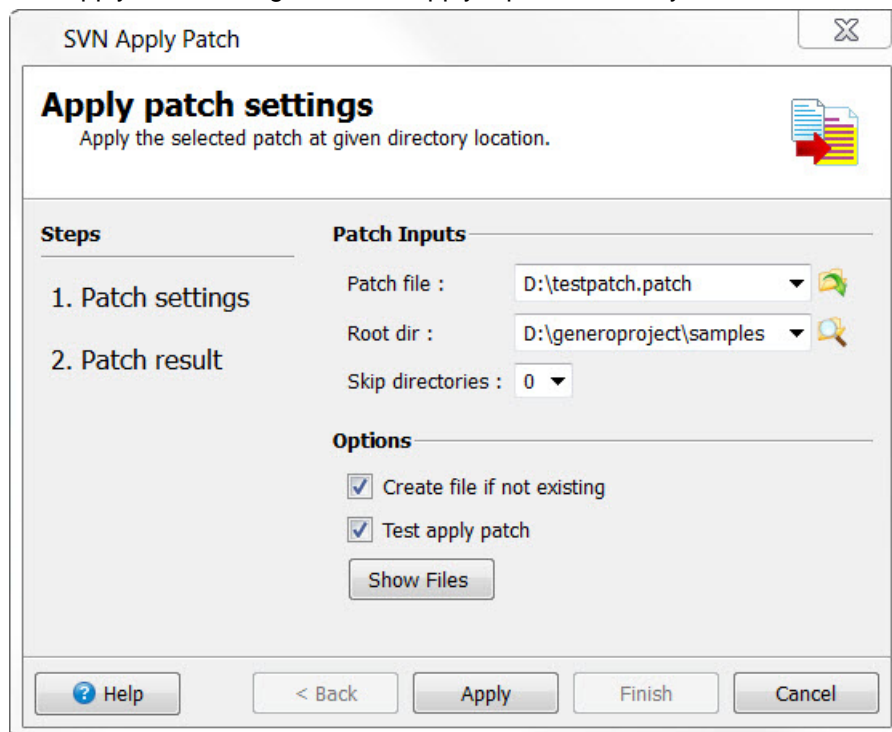


Figure 302: Apply Patch dialog

Patch file

The path of the selected patch file.

Root dir

The path of the directory where the patch is to be applied.

Skip directories

Integer value defining the directory level to be search for the file to be patched.

Create file if not existing

If the file to be patched is not present in the current directory path, the corresponding file will be created as an unversioned file. If the file is unchecked in the [SVN Status view](#) on page 543, the corresponding file will not be created.

Test apply patch

Displays the output of the **Apply Patch** command without actually applying it to the files; files will not be modified.

Show files

Opens a dialog displaying a list of all the files that will be patched.

Merge/Revert dialog

The SVN Merge/Revert dialog lets you merge or revert revisions.

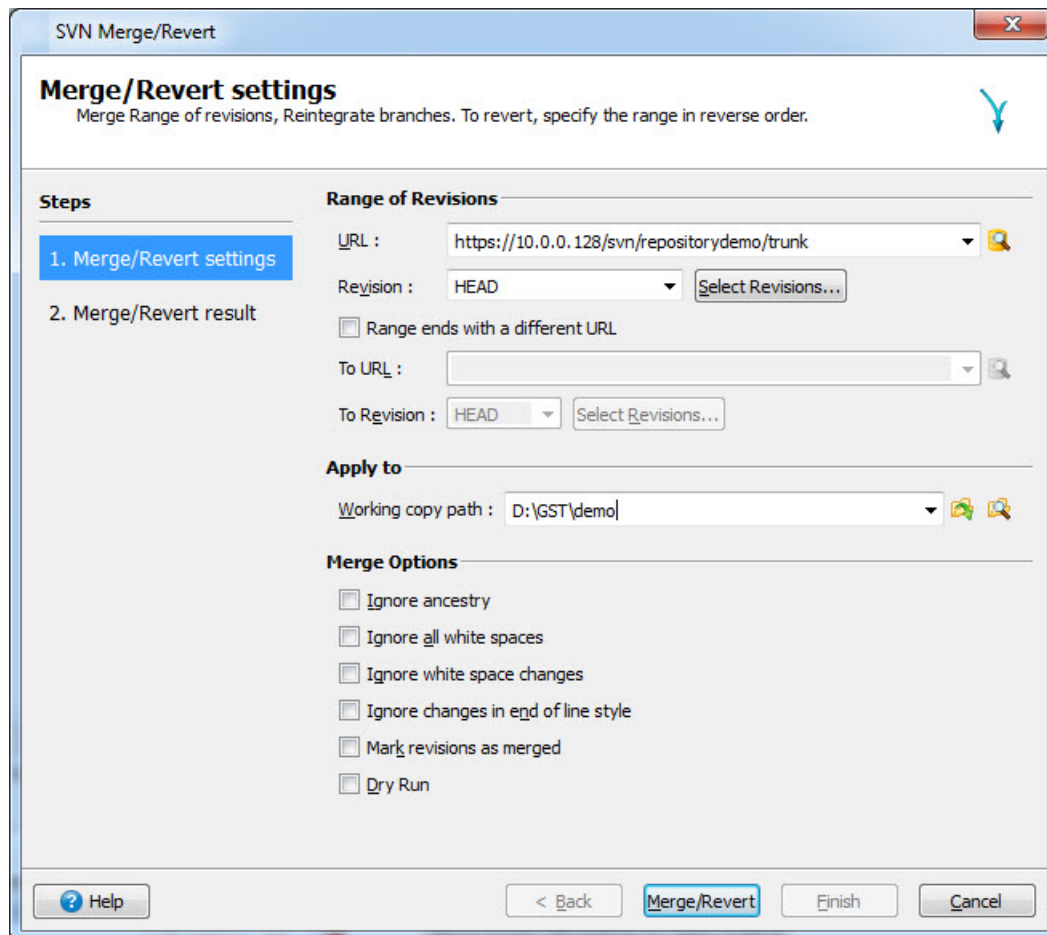


Figure 303: SVN Merge dialog

- **URL** - Enter the URL of the repository and select the Revision number, or HEAD for the most recent revision
- **To URL** - Enter the URL of the repository and Revision number at which you want to end, or click the **Use the start URL** button to continue to the end.

To access the dialog:

- Select **SCM >> Merge/Revert** from the Genero Studio menu.
- Right-click on a file and select **SCM >> Merge/Revert** from the context menu.
- Right-click on a revision entry from the SVN Log view and select either Revert changes from selected revisions OR Revert to this revision

You can revert the changes from a single revision, or you can revert the changes across a range of revisions.

The SVN Copy dialog

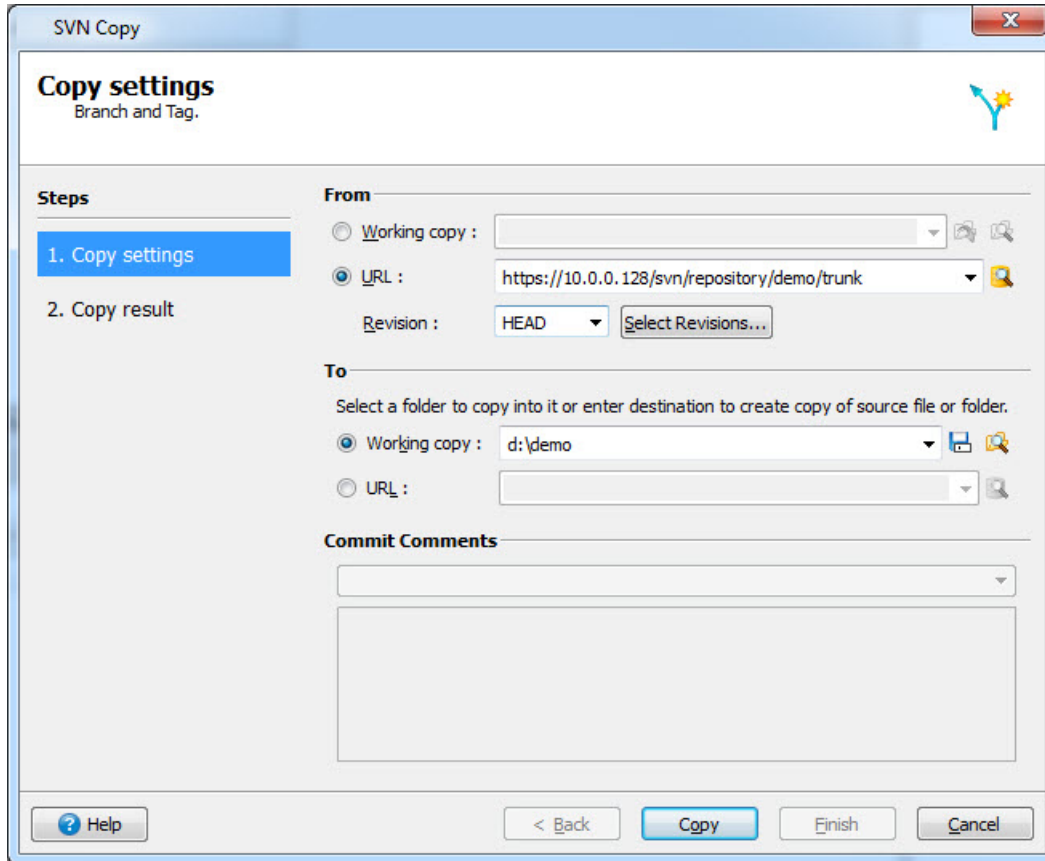


Figure 304: SVN Copy dialog

Select the corresponding radio buttons (Working Copy/URL) to indicate that you will:

- Copy the working copy of the project file from which you invoked the SVN copy dialog to a different repository having the URL that you have entered. Select the revision (HEAD is the default).
- Copy the selected file from a repository having the URL that you have entered, to a working copy having the location that you have entered. Select the revision (HEAD is the default).
- Copy the selected file from one repository to another. Select the revision (HEAD is the default).

The Copy result is shown.

SVN Checkout dialog

The SVN Checkout dialog guides you through the checkout process, allowing you to load copies of repository files into a checkout directory on your local system:

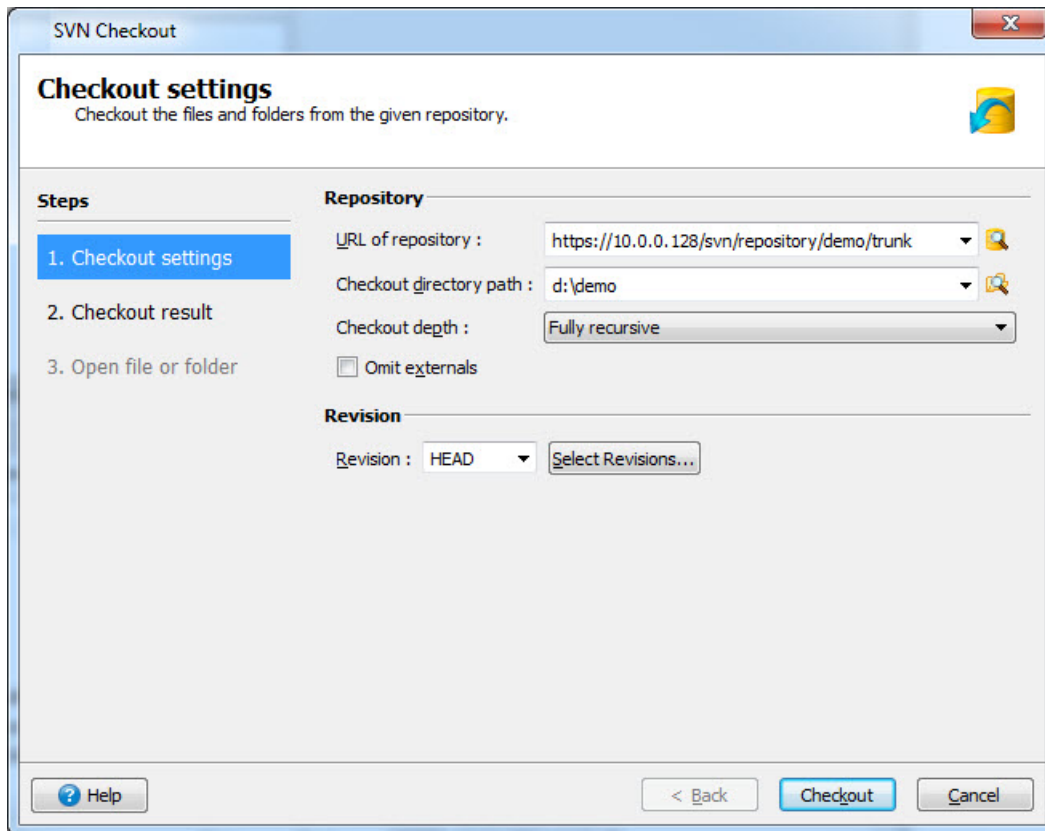


Figure 305: SVN Checkout dialog

The **Repository** section:

- **URL of repository** - enter the URL of the SVN repository that you wish to access.
- **Checkout directory path** - enter the path of the directory for the checked out files; the Browse button allows you to locate the path.
- **Checkout depth** - choose the level you desire; for example, "only file children" will checkout the files but not the directories.
- **Omit externals** - check to exclude "external items".

The **Revision** section:

Specify the particular version from which to checkout the files. The Select Revisions button opens the [SVN Log Select revision dialog](#).

After the Checkout button is clicked

Messages related to the checkout operation display in the **Output View**.

The results of the Checkout and other SCM commands are displayed in **Results** pages. A **Run in background** button, at the bottom of the SCM dialogs, allows you to close the dialog but continue running the process in the background, as indicated in the Task Manager.

SVN Lock dialog

The SVN Lock dialog is used to lock a file..

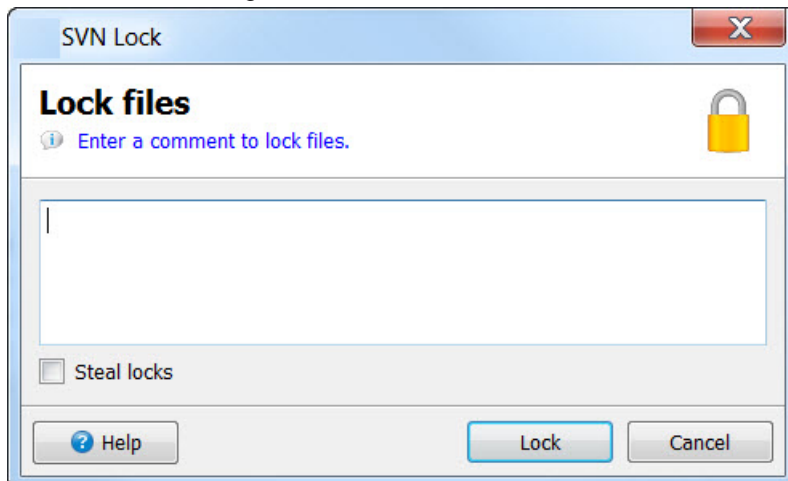


Figure 306: SVN Lock dialog

Comment

Enter a comment to lock the file.

Steal locks

Forcefully acquire the lock if locked by another user.

SVN Properties dialog

Use **SCM >> Properties** from the right-click context menu on a file to make changes to SVN properties. Use the integrated Toolbar to add, modify, or delete properties. See your *Subversion* manual for additional information about properties.

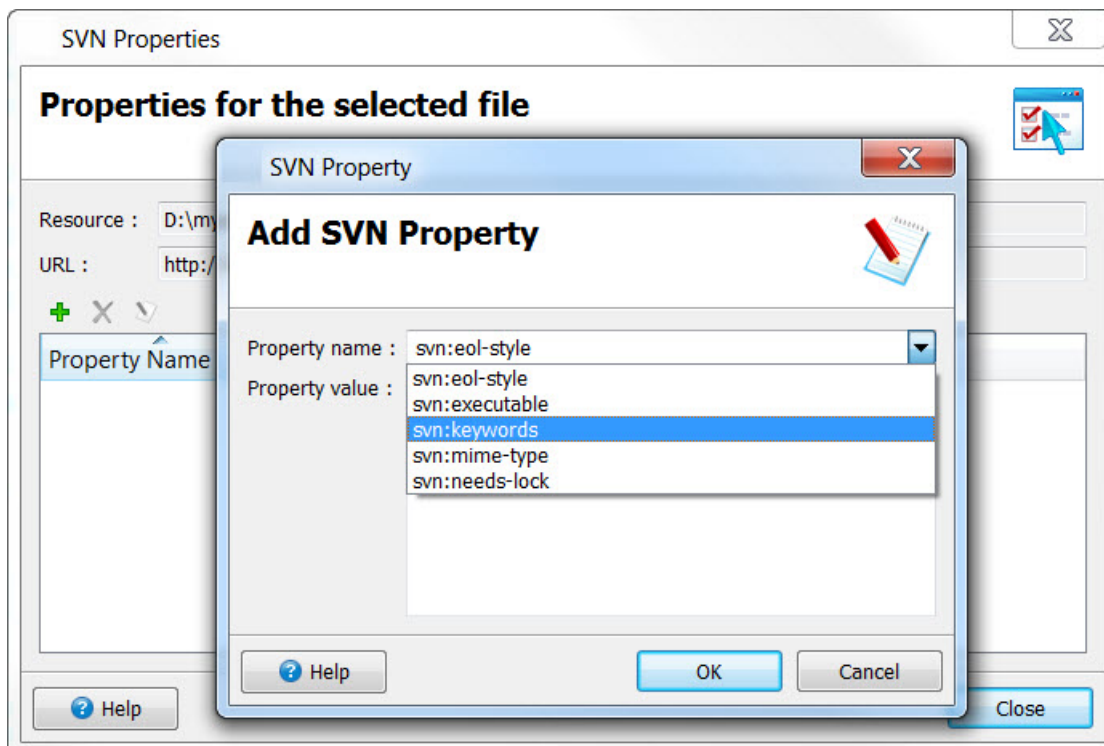


Figure 307: SVN Properties dialog

SVN Repository view

The SVN Repository displays the contents of the selected repository and version.

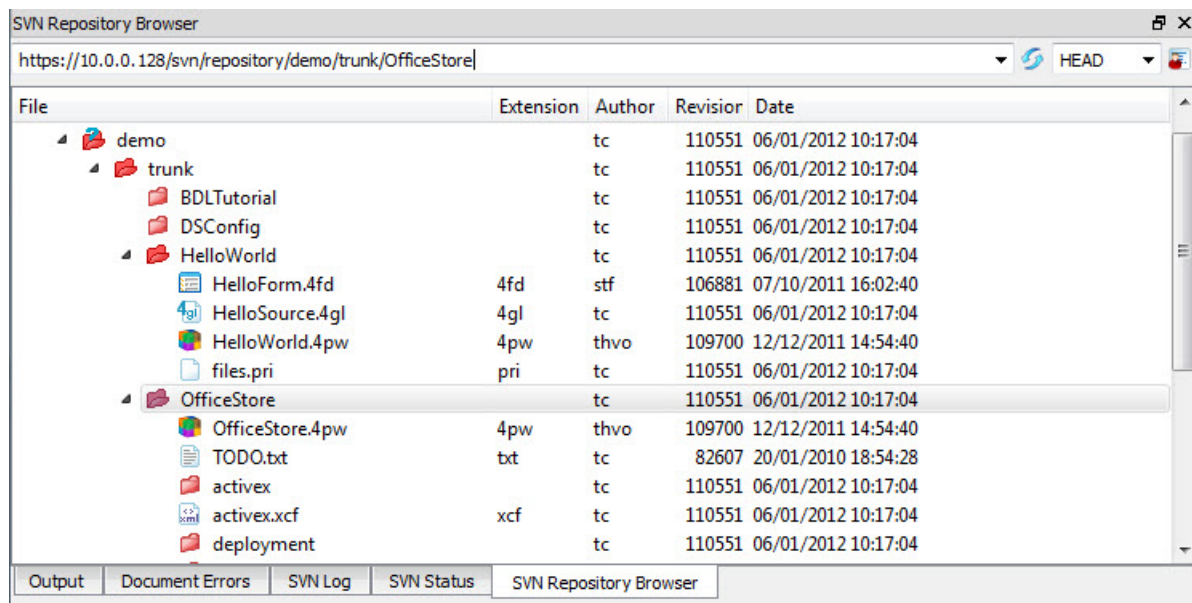


Figure 308: SVN Repository Browser

When a user does not have the rights to access a directory in the repository, the directory is marked with a forbidden icon



Specify a Subversion client

To specify the location of your SVN client, select **Tools >> Preferences >> Source Code Management >> Subversion** from the main menu.

1. Select **Tools >> Preferences >> Source Code Management >> Subversion** from the main menu.
2. In the SVN client path, enter or browse to the path of your SVN client; include the bin directory in this path.
3. Press **Check Version** to validate the SVN client and to show SVN version details.

SVN error messages

A list of SVN error messages. For messages that are not self-explanatory, additional information is provided.

Important: It is likely that the error is an SVN error. When Genero Studio executes an SVN command, it asks the SVN executable to perform the command. Genero Studio displays the result, success or error. You will likely have to refer to the [official SVN documentation](#) to identify the problem and find the solution.

Table 153: SVN Error Messages

Number	Description
GS-22001	SVN Tool error.
GS-22003	Cannot start SVN Tool. The SVN Tool is not properly set in preferences.

Number	Description						
	In Tools >> Preferences, Source Code Management, Subversion , set and check the SVN client path.						
GS-22004	SVN Add command failed.						
GS-22005	SVN Commit command failed.						
GS-22006	SVN Export command failed.						
GS-22007	SVN import command failed.						
GS-22008	SVN Info command failed.						
GS-22009	SVN mkdir command failed.						
GS-22010	SVN remove command failed.						
GS-22011	SVN rename command failed.						
GS-22012	SVN rename URL command failed.						
GS-22013	SVN resolved command failed.						
GS-22014	SVN revert command failed.						
GS-22015	SVN log command failed.						
GS-22016	SVN status command failed.						
GS-22017	SVN update command failed.						
GS-22018	Apply patch failed.						
GS-22019	<p>Server certification error.</p> <p>In the displayed dialog, selection an option:</p> <table border="0" data-bbox="440 1144 1464 1375"> <tr> <td data-bbox="440 1144 941 1207">Accept permanently</td> <td data-bbox="950 1144 1464 1207">You will not be asked for the server certificate for the same repository again.</td> </tr> <tr> <td data-bbox="440 1218 941 1323">Accept Once</td> <td data-bbox="950 1218 1464 1323">Accepts the certificate only for the current operation. For the next operation, this dialog will appear again.</td> </tr> <tr> <td data-bbox="440 1333 941 1375">Reject</td> <td data-bbox="950 1333 1464 1375">Cancels the operation.</td> </tr> </table>	Accept permanently	You will not be asked for the server certificate for the same repository again.	Accept Once	Accepts the certificate only for the current operation. For the next operation, this dialog will appear again.	Reject	Cancels the operation.
Accept permanently	You will not be asked for the server certificate for the same repository again.						
Accept Once	Accepts the certificate only for the current operation. For the next operation, this dialog will appear again.						
Reject	Cancels the operation.						
GS-22020	<p>SVN Authentication failed error.</p> <p>The current user does not have access to the SVN repository.</p> <p>Provide a valid user name and password in the authentication dialog box.</p>						
GS-22021	SVN proplist command failed.						
GS-22022	SVN Copy command failed.						
GS-22051	<p>Item found in checkout.</p> <p>Double click to open.</p>						

Report Writer

The Genero Report Writer includes a graphical report designer, report engine, and report viewer. Report applications are written using the Genero Business Development Language.

Important: This feature is not supported on mobile platforms.

- [grw_section_getting_started.ditamap](#)
- [Create a report program](#) on page 561
- [Create the data schema](#) on page 655
- [Create a report design document](#) on page 656
- [Report templates](#) on page 840

Get Started with Reports

These topics give you the information you need to initially work with Genero Report Writer.

- [Introduction to Reports](#) on page 551
- [The Reports demo](#) on page 555
- [Configure fonts and printers](#) on page 559

Introduction to Reports

Genero Report Writer provides a means for creating high-quality reports.

- [What is Genero Report Writer \(GRW\)?](#) on page 551
- [GRW runtime architecture](#)
- [Steps to a Report](#)

What is Genero Report Writer (GRW)?

Genero Report Writer provides a GUI interface to design reports that are based on data typically retrieved from a database; the layout of the report is specified using the drag-and-drop report designer. Gathering the data for the report is separated from the layout, enhancing security, allowing reuse, and providing an interface that protects the report from changes to the actual data storage.

Report Creation

- You write an application to retrieve the data; the data may come from a database, from a data file, from an XML file, or from a Web service. This application retrieves the report data and outputs it to Genero Report Writer. The application can be coded by hand, or can be generated using the Business Application Modeler.
- Once the application is created, you use a tool to extract the data schema for the report. The data schema is used to populate the Data View in the [Genero Report Designer](#). From the Data View, data objects can be dragged and dropped onto the design page.
- You [design the report layout](#) using Genero Report Designer, dragging and dropping the report elements (including data objects) onto the report design page. This creates a [Report Design Document](#) (4rp file).
- You can specify the default report output options using the Report Designer menu, or you can use language-specific APIs in your application to modify the output options.
- End-to-end streaming (see [How It Works](#)) from the Genero DVM to the [Genero Report Viewer](#) provides "print-as-you-go" processing; the data is produced on the fly.

The Report Design Document

The Report Design Document (4rp) specifies the appearance of the report and its data:

- Report elements (containers (such as [Mini Page](#) and [Horizontal Box](#)), and their child elements (such as [Word Box](#), [Image Box](#), and [data objects](#)) can be dragged and dropped onto the report page and rearranged.
- Properties in the [Properties View](#) describe the report element. The default values for element properties can be changed, or calculated using [expressions](#). An expression can be used to change the appearance of the element, or change it conditionally; for example, an expression might turn the background color red if the value for the report element is greater than 1000.
- The report design elements can contain values that are based on other values in the report; for example, totals and subtotals.
- Various Report Design Documents can be created using the data schema from a single 4gl file.
- A Report Design Document can be output in different formats.

See [Designing a Report](#).

The Report Output

A Genero report is based on XML standards (XML reports, XML data, SVG output). Since the data is in XML format, you can have multiple views of the same data. The report can be output to various formats, or directly to a printer or other output devices.

You can use Genero Report Writer to create reports for a variety of purposes. For example, you can produce:

- Lists
- Reports with a customized format, such as invoices, corporate documents, or accounting reports
- Reports that include images
- Reports on pre-printed forms
- Labels
- Business Graphs (such as MapCharts, Category Charts, and XY Charts)
- XML documents
- HTML documents
- Excel spreadsheets
- Documents in Microsoft™ RTF format
- ASCII-based reports, to print legacy Genero reports "as is"

The rapid delivery of pages, with no need for temporary files or temporary tables, consumes less memory and speeds up the printing of very large reports.

See [Configuring Fonts and Printers](#) for tips about printers, fonts, and custom paper sources.

GRW runtime architecture

Information about the Genero components used to produce a report from Genero Report Writer.

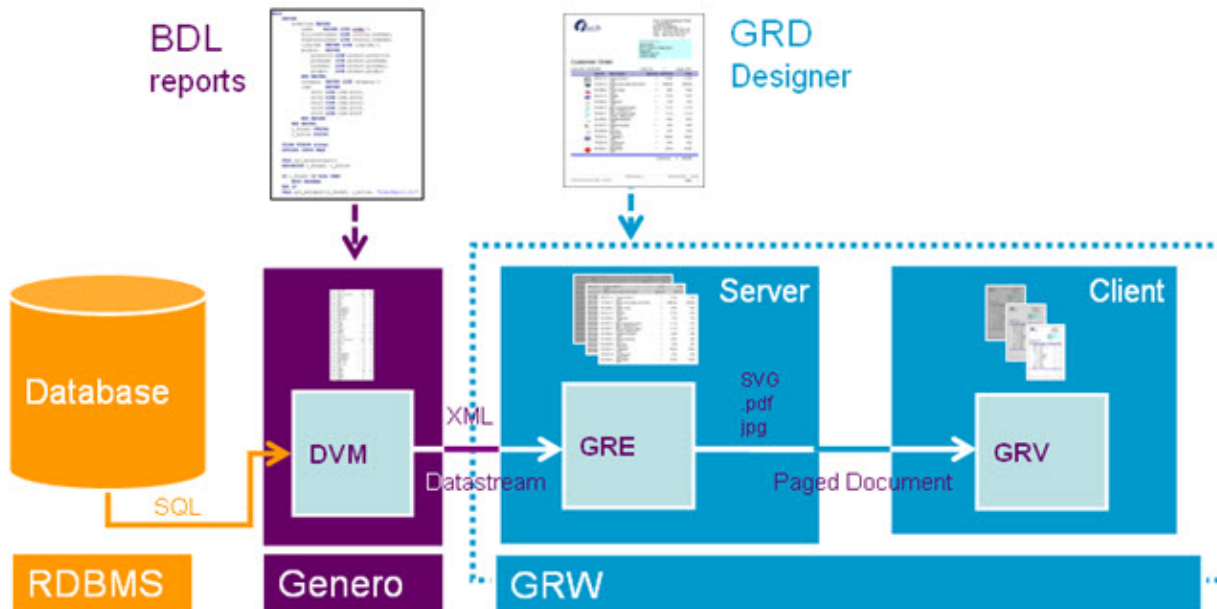


Figure 309: Genero Report Writer Workflow

- The [Genero Business Development Language \(BDL\)](#) defines the data needed from the database
 - **The Genero DVM (Dynamic Virtual Machine)** executes the BDL code to retrieve the data
 - **The Genero DVM** streams data from the database to the **Genero Report Engine (GRE)**
- The [Genero Report Designer \(GRD\)](#) graphically lays out the report
 - **The Genero Report Engine (GRE)** lays out paged streams to **Genero Report Viewer (GRV)**
- The [Genero Report Viewer \(GRV\)](#) (part of the **Genero Desktop Client**) displays the report on the client; the [Genero Report Viewer for HTML5 on page 558](#) displays the report in a browser; alternatively, the report can be displayed as a PDF or Image, as an HTML or XML file, or output to an Excel spreadsheet.

Steps to a Report

To use Genero Report Writer to create a report, you need to properly prepare your environment, create the report application and report, and run and test the report.

- [Set the Report Writer environment](#) on page 553
- [Create a Report](#) on page 554
- [Run the Report](#) on page 554

Set the Report Writer environment

Complete these steps to set up the Report Writer environment

- From the Genero Studio main menu, select **Tools>>Configurations**.
- In the **Compile/Runtime** section of the Genero Configuration Management window, the default environment sets are listed.
- The **Report Writer** environment set contains the variables that must be set for Genero Report Writer. This environment set should be active (has a check mark in the box).

Create a Report

Creating a report involves setting up a project to store the report files, developing a report application, designing the report, and testing the report.

Set the Report Writer environment. See [Set the Report Writer environment](#) on page 553.

This procedure is provided to give you a quick start, or overview, for creating a report application. For each step, there are many options that are not discussed here, but are covered in other topics. Use this procedure as a guide for using Genero Report Writer to create reports.

1. Create the project.

A project contains the nodes for storing and managing your report application files. While there are a variety of ways to create a new project

- a) Select **File >> New, Genero Files, Simple Project (.4pw)** and click **OK**.
- b) Select **File >> Save as** and complete the dialog to save your project.

2. Set up the project's application node.

The project tree should have a Group node, and the Group node should contain an application node, a databases node, and a library node.

- a) Rename the nodes to meaningful names.
- b) In your application node, create two virtual folders named "src" and "reports".

These virtual folders are simply a suggestion; you are free to organize your files as you see fit.

3. Create your report application.

The Genero report application (4g1) is responsible for reading data from a database (or alternate data source) and streaming it to the Genero Report Engine. See [Writing the Genero BDL report program](#) on page 564.

4. Generate the data schema.

The data schema (rdd) file identifies the fields and the grouping of data streamed by the Genero reporting application to the Genero Report Engine. It is used by the report design document, giving the designer the ability to select fields for placement on a report and setting report triggers. See [Generate a data schema from a Genero BDL report program](#) on page 572.

5. Create a report design document.

A report design document (4rp) is a file that defines a single report. For each report you need, you would create a separate report design document. When it comes time to create a specific report, you tell the report application which file to use.

- a) Select **File >> New, Reports** and select either **Empty Report (.4rp)** or **List Report (.4rp)** to start a new report.
- b) On the Data View tab, select the data schema (rdd).
- c) Report output can be output in various formats. Select **File>>Report properties >> Output configuration** to change the default output for the report.
- d) Save the report design document and add it to the reports virtual folder in your application node.

For more information on working with the report design document, see [The Report Design Document](#) on page 657.

6. Build the report application.

Right-click the report application node or the report group node in the Genero Studio Projects tab and select **Build** to compile and link the application.

Run the Report

- Right-click the report application node in the Project Manager and select **Execute** to run the program.

Note: If the report data will come directly from a database, your BDL program must have access at execution to the database where the data is stored.

- By default a new report is in SVG format, and the output is set to Preview using the [Genero Report Viewer](#); you can change these defaults in your Report Design document, or in your BDL program code.

The Reports demo

The OrderReport demo application provides a sample reporting application, along with various reports design documents.

These topics provide an overview of the general functionality of the demo application, as well as a description of the provided sample reports.

- [The Reports demo overview](#) on page 555
- [Run the Reports demo](#) on page 555
- [Genero Report Viewer for HTML5](#) on page 558

The Reports demo overview

When Genero Studio launches, the Welcome Page lists the demo projects. Select **Reports** to open the OrderReport demo project in Genero Studio.

The Reports demo project consists of:

- Application files, generated or written using Genero Business Development Language.
- A set of report definition files, created with the Genero Report Designer.
- The data schema for the database used with the demo reports.

The demo project files can be found in `Documents and Settings\<username>\Documents\My Genero Files\samples\Reports`. Each project is organized by an appropriately-named directory.

To run the demo:

- Expand the **Reports** project in the Project Manager, and right click on the **Order Report** application node.
- Select **Execute** from the context menu.

Run the Reports demo

When the report begins to run, the application displays a dialog box to allow you to select the report you wish to run, and the output format.

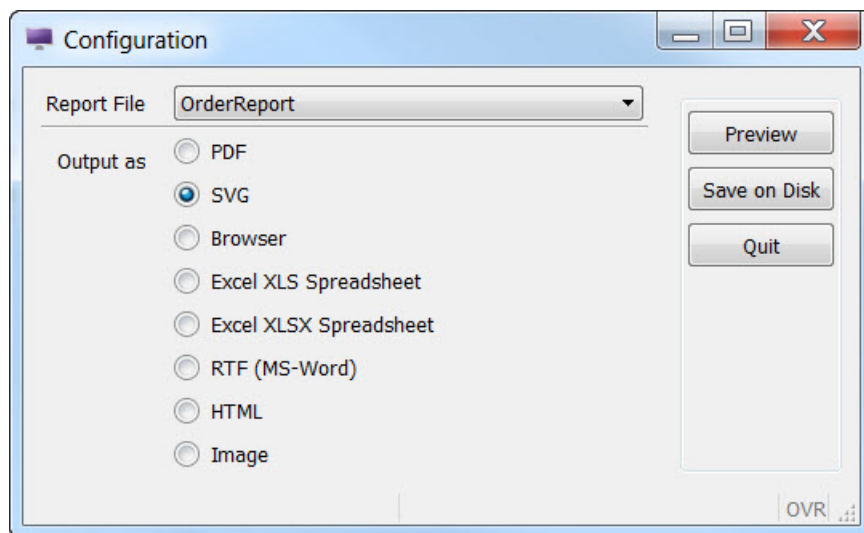


Figure 310: Reports demo form

1. Select a report from the combobox. See [Reports demo reports](#) on page 556 for the list of reports.
2. Select the output format. See [Reports demo outputs](#) on page 557 for the output options.
3. Select the desired action, to **Save On Disk** or **Preview**. See [Reports demo outputs](#) on page 557 for more details.

Reports demo reports

When you run the Reports demo, you select the desired output.

With the Reports demo, you select a report from the combobox. The reports include:

OrderReport	A report detailing a customer order. This report illustrates the use of the section property, as well as the use of different page headers and footers on each page.
OrderLabels	A report consisting of pages of mailing labels for customers. This report illustrates the labels functionality, enabling you to print multiple labels on a single report page.
OrderList	A report showing a text-based list, based on the list template.
OrderStock	A report displaying stock information, including a bar code. This report illustrates the use of attachment and positioning rules, and shows you how to design a report onto pre-printed paper using a background image.
ListDemo	a report in list format that includes images, totals, and conditional text coloring. This report provides two examples of RTL expressions that control the font color and the "thumbs-up/thumbs-down" image at runtime, based on a field value.
Generic List	A report that provides an interface that allows you to select which fields are included in the list report.
TableDemo	A report in table format. This report shows a simple usage of a table.
GroupTableDemo	A report in table format. This report shows a table with several header and body rows.
CategoryChart*	Several examples showing a category chart, or a chart that is grouped by two fields. These examples show how different charts can be produced from the same data
MapChart*	Several examples showing a map chart, or a chart that is grouped by one field. These examples show how different charts can be produced from the same data
StaticPivotTable	A table of customer data, grouped by customers and orders. Note: Static pivot tables are not yet supported in Java.
DynamicPivotTable	Allows you to select which fields to use as dimensions (for grouping) and which fields to include as measures. Generates a table of user-selected measures, grouped by user-selected dimensions.
MasterReport	A report showing the use of sub-reports.

Note: Sub-reports are not yet supported in Java.

Reports demo outputs

When you run the Reports demo, you determine the format of the report output, and whether it is saved to disk.

Output formats

The Output As section of the Reports user interface allows you to select the output format for your report.

Image	<p>An image file is created. By default, with the demo app, it is created in .jpg format.</p> <ul style="list-style-type: none"> • If preview is selected, the image displays in the default image viewer for the client. • If save on disk is selected, the image is created in the GRW demo project directory.
PDF	<p>A .pdf file will be created in the GRW demo project directory. The file can be viewed with an Acrobat PDF reader.</p>
SVG	<p>An SVG file will be created in the GRW demo project directory.</p>
Browser	<p>An SVG file will be created for viewing and is displayed using HTML5, with the Genero Report Viewer for HTML5 on page 558.</p>
Excel XLS Spreadsheet	<p>The report data is output in Excel format.</p>
RTF (MS-Word)	<p>The file is save in Rich Text Format (RTF), which provides a format for text and graphics interchange that can be used with different output devices, operating environments, and operating systems; more that just Microsoft Word.</p>
HTML	<p>The report document is output in HTML format</p>

Output actions

You can either view the report immediately, or you can save the report to disk.

Preview	<p>Displays the report according to the selected output option. A copy is not saved to disk.</p>
Save On Disk	<p>Saves a copy of the report to disk. The report will be saved in the demo project directory. The name of the report will be "report", with an extension of .pdf or .svg, or image000#.jpg where # is incremented sequentially from 1 for each successive image that is saved to disk.</p>

Genero Report Viewer

If you choose **SVG** as the output format and **Preview** as the action, the report displays in the Genero Report Viewer, which is automatically installed as part of the Genero Desktop Client.

The Report Viewer is a component of the Genero Desktop Client that is automatically installed by the installation program of the GDC; it can be used to display reports that are in the SVG format only.

Important: SVG format for reports is currently not supported for previewing when running with the Web Client.

Item ID	Description	Quantity	Unit Price	Total
Order 5 from 09/09/2008				
Partial total of order 5.0 from 09/09/2008 :				728.7
FU-004-A	Grandfather clock Unit	1.00	122.00	122.00
FU-008-A	Office chair Unit	1.00	129.00	129.00
FU-011-A	Table lamp Unit	1.00	27.00	27.00
SU-001-A	Blue and green pens Blue - Unit	1.00	1.20	1.20
SU-001-D	Blue and green pens Green - Unit	1.00	1.20	1.20
SU-004-A	Clipboard Unit	1.00	7.20	7.20
SU-005-A	Coloring pencils Blue - Unit	1.00	0.15	0.15
SU-005-C	Coloring pencils Red - Unit	1.00	0.15	0.15
SU-005-B	Coloring pencils Yellow - Unit	1.00	0.15	0.15
SU-006-A	Magnifying glass Unit	1.00	9.90	9.90
SU-002-A	Notebook Unit	1.00	4.00	4.00
SU-009-A	Organizer book Unit	1.00	4.00	4.00
SU-008-A	Paper clasp Unit	1.00	0.80	0.80
SU-010-A	Pen Unit	1.00	3.50	3.50
SU-003-A	Scissors	1.00	1.35	1.35

Figure 311: Genero Report Viewer

Use the **File>>Print** menu option in Report Viewer to send the report to a printer.

Genero Report Viewer for HTML5

If you choose **Browser** as the output format and **Preview** as the output action, the report displays in the Genero Report Viewer for HTML5

Genero Report Viewer for HTML5 is a lightweight report viewer based on pure HTML5 and JavaScript technologies. It is optimized for low bandwidth and slow networks. The rendering is pixel-exact. Fonts are optimized, including good performance for Asian fonts. In terms of performance, it allows streaming and can display pages immediately as they become available. It fetches only what is necessary. The reports are static files and, as such, can be viewed at any time by sharing the relevant URLs.

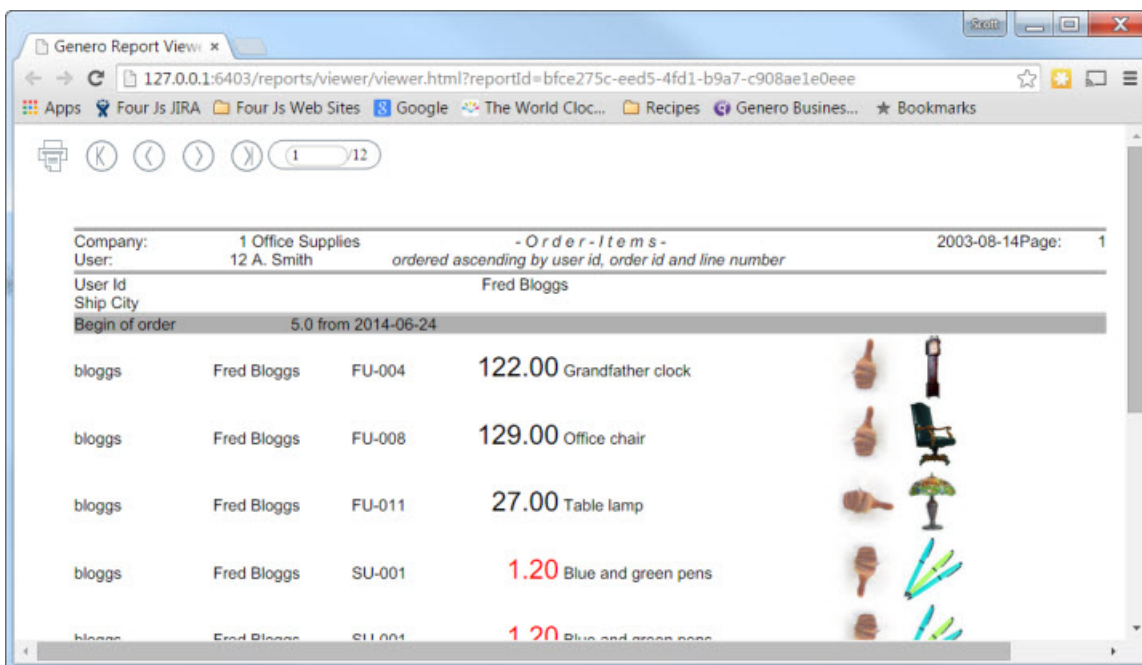


Figure 312: Genero Report Viewer

In addition to displaying the report, the report viewer provides navigation options. It supports direct navigation to a specific page. The Print icon prepares the report in a single HTML page that you can then print using your browser's print functionality.

Configure fonts and printers

To consistently output well-formatted reports, you should verify that you have the necessary fonts and printers available.

- [Available Fonts and Printers](#)
- [Tips on installing common Fonts or new Fonts](#)
- [Specific Font Types](#)

Available fonts and printers

Scripts are provided to get information about the available fonts and printers for that machine.

The scripts can be found in the in the `$GREDIR/bin` directory. The scripts are executed from the command line.

- `fontinfo` - Lists the fonts available on your server.
- `fontinfopdf` - Lists only the fonts that can be used in PDF files.
- `printerinfo` - Lists the available printers.

For additional information about the use of `printerinfo`, see [Support for custom paper sources](#) on page 591.

What printers are supported?

The `printerinfo` script lists the available printers, however not all available printers are supported.

The printers supported must be graphical printers.

- On Windows™, a Windows™ printer driver for the printer in question must be installed.
- On UNIX™, the printer must support Postscript or a converter from Postscript to the printer's native language (typically a ghostscript backend) must be installed and configured.

Plain text printers or bar code printers with proprietary control languages are not supported.

Search the internet for Windows™ drivers for some brands of thermal transfer bar code printers when the vendor does not provide a driver, as others have specialized in providing such drivers.

Do not use dot matrix printers in graphical mode, even if the drivers exist. The speed is typically too slow, and the print results are poor.

UNIX™ and CUPS

- CUPS should be used if available. Direct the browser to `http://host:631`. Always make use of the "print the test page" option during the installation.
- If CUPS is not available and the printer is listed in `printerinfo` but does not work, test if any other graphical application (such as OpenOffice or FireFox) can print to that printer.

Install new or common fonts

The best results are achieved if the fonts on the server where Genero Report Engine (GRE) is installed and those used in the Genero Report Designer (GRD) are the same.

- If GRE and GRD are installed on the same machine, there is no problem.
- If GRE and GRD are installed on different machines that have the same operating system and the same installed fonts, there is no problem.

On Windows™ systems you can use [scripts](#) to get information about the available fonts.

A report that uses only a few positioned items will still lay out correctly, however, even if the fonts used in the Designer and the fonts used in the GRE server differ:

- If the server cannot find a specified font, it does not raise an error; it uses a fallback font instead. A warning has been added to the runtime system in debugging mode (`$GREDEBUG > 0`) when a fallback font is used because a font specified in a template file cannot be found on the system.
- If no font is specified, a default font is used.

Specify a font at the root of the document, to avoid potentially changing output when a new version of Java™, or a different server, is used. In the Report Designer, set the font property for the Page Root node.

Not all fonts contain all possible characters. Some fonts will not contain certain glyphs. In this case, GRE will attempt to take the missing characters from a different font that contains the glyphs. For example, the monospaced (fixed-width) font "Courier" does not contain graphics characters. If a report contains a grid that is drawn using graphics characters, GRE might substitute characters from a set that is not fixed-width, causing the layout to break. Avoid this problem by using a font like "Lucida Sans Typewriter", which is both fixed-width and contains the required characters.

Not all fonts can be used for embedding in PDF: the license flag contained in the font might prevent a font from being used. The utility `$GREDIR/bin/fontinfopdf` lists all fonts that can be used in PDF documents. While True Type fonts generally work, sometimes a Type 1 font will not appear in the list because it is available only as text (.pfa) but not as a binary file (.pfb). In this case, the binary font can be created by using font compilation tools such as `pfa2pfb` or `t1binary` from the `tlbinaries` package.

Specific font types

Determine whether the fonts you wish to use are available for your system, and are usable by both the designer and the runtime system.

Type 1 and Windows™ TrueType Fonts

Genero Report Engine is capable of reading both Type 1 and Windows™ TrueType fonts. It is possible to copy fonts from Windows™ to UNIX™. Since the TrueType directories can differ between Linux™ distributions and between different UNIX™ versions, copy the fonts into `/usr/lib/X11/fonts/TTF` (for TTF fonts) or to `/usr/lib/X11/fonts/Type1` (for Type1 fonts), where they will be found by GRE. To check whether GRE sees the font, run the executable `$GREDIR/bin/fontinfo`, listing the fonts seen by the GRE.

Important: You must identify whether the fonts you wish to copy violate any copyrights.

Lucida family of Fonts

SUN-Java contains a basic set of fonts (Serif, Sans Serif and Monospaced) in the "Lucida" family. Some distributions of the Java™ Runtime Environment do not contain all fonts, but it is legal to copy the fonts from one distribution to another. These fonts contain a large part of the Unicode characters.

The available codes are listed in <http://java.sun.com/j2se/1.5.0/docs/guide/intl/font.html#lucida>.

Liberation fonts

Another option for free fonts is the "Liberation" fonts originally provided by Red-Hat. These fonts have the same metrics (character width) as the Microsoft™ fonts Arial, Times New Roman, and Courier, and a similar look. These fonts can be installed on both Windows™ and Linux™. A description of the fonts can be found at http://en.wikipedia.org/wiki/Liberation_fonts. The fonts can be downloaded at <https://www.redhat.com/promo/fonts>.

Asian Fonts

When using Asian fonts in PDF or SVG documents:

- Set the `fidelity` property to "true" on any `WORDBOX` or `WORDWRAPBOX` using Asian characters.
- For reports running in compatibility, set the parameter `fidelity` to "true" in calls to the API function `fgl_report_configureCompatibilityOutput()`.

Make sure the specified fonts contain the required characters. The designer will display the characters correctly even though the selected font may not contain them; the runtime system does not have this behavior.

Create a report program

With Genero Report Writer, you create the code to fetch and stream the data to the Genero Report Engine, to select the report design document, and to influence how the report is output, overriding default output settings or output settings specified in the report design document.

Genero Report Writer provides the ability to generate or write the data source, which defines the data that is to be streamed to the report engine.

Two tools exist to assist you in generating your Genero report application:

- The [Report Data Wizard](#) provides an easy-to-use interface for creating the data source.
- The [Business Application Modeler](#) allows you to diagram an application, of which the report is one component.

Once defined, a data source can be integrated into the code of your existing applications, or it can be included in a new custom reporting application. The use of the Genero Report Writer APIs allow further control over the processing of the reports.

The language you use to create your data source will likely be dependent upon the language used for your existing applications, as the goal is to have seamless integration of the Genero Report Writer code with these applications.

- [Genero BDL and the Report Writer](#) on page 562

Genero BDL and the Report Writer

You can create a data source and control the report output with a program written in Genero.

The runtime architecture (GRW for Genero BDL)

The Genero BDL application is one part of the runtime architecture for Genero Report Writer for Genero Business Development Language.

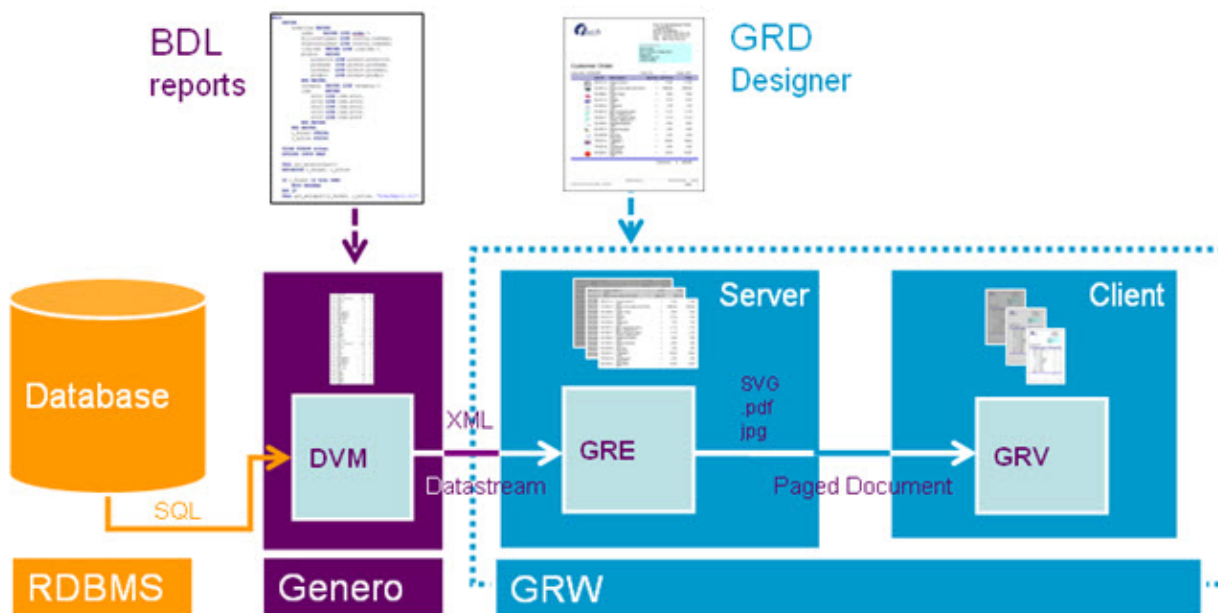


Figure 313: Genero Report Writer workflow

- The [Genero Business Development Language \(BDL\)](#) defines the data needed from the database
 - **The Genero DVM (Dynamic Virtual Machine)** executes the BDL code to retrieve the data
 - **The Genero DVM** streams data from the database to the **Genero Report Engine (GRE)**
- The [Genero Report Designer \(GRD\)](#) graphically lays out the report
 - **The Genero Report Engine (GRE)** lays out paged streams to **Genero Report Viewer (GRV)**
- The [Genero Report Viewer \(GRV\)](#) (part of the **Genero Desktop Client**) displays the report on the client; alternatively, the report can be displayed as a PDF or Image, as an HTML or XML file, or output to an Excel spreadsheet.

Steps to a Report (GRW for BDL)

To use Genero Report Writer to create a report, you need to properly prepare your environment, create the report application and report, and run and test the report.

- [Set the Report Writer environment \(GRW for BDL\)](#) on page 562
- [Create a Report \(GRW for Genero BDL\)](#) on page 563
- [Run the Report \(GRW for BDL\)](#) on page 564

Set the Report Writer environment (GRW for BDL)

- Select **Tools >> Genero Configurations**. The **Genero Configuration Management** window opens.
- The default environment sets are listed in the **Environment Sets** section.
- The **Report Writer** environment set contains the environment variables that must be set for Genero Report Writer. This environment set should be active (checked).

Note: At the application level, you must include the library `libgre.42x`. See [Create a Report \(GRW for Genero BDL\)](#) on page 563.

Create a Report (GRW for Genero BDL)

Creating a report involves setting up a project to store the report files, developing a report application, designing the report, and testing the report.

Before you begin, set the Report Writer environment. See [Set the Report Writer environment \(GRW for BDL\)](#) on page 562.

This procedure is provided to give you a quick start, or overview, for creating a report application. For each step, there are many options that are not discussed here, but are covered in other topics. Use this procedure as a guide for using Genero Report Writer to create reports.

1. Create the project.

A project contains the nodes for storing and managing your report application files.

- a) Select **File >> New**.
- b) In the **Categories** listing, select the language you wish to work with.
- c) In the **Types** listing, select the default project (4pw) and click **OK**.
- d) Select **File >> Save as** and complete the dialog to save your project.

2. Set up the project's Application node.

The project tree should have a Group node, and the Group node should contain an application node, a databases node, and a library node.

- a) Rename the nodes to meaningful names.
- b) Right-click on the application node to create files and folders. For example, create two virtual folders named **src** and **Designs**, and create a source file named `myreport.4gl`.

Note: You must add the `END` and `END MAIN` code to the source file before it enables you to save.

- c) In the **Properties** view, set the **External dependencies** value to `libgre.42x`.

For more information about the `libgre.42x` library file, see the [GRW reference for Genero BDL applications](#) on page 599 section.

3. Create your report application.

The report application is responsible for reading data from a database (or alternate data source) and streaming it to the Genero Report Engine.

4. Generate the data schema.

The data schema file identifies the fields and the grouping of data streamed by the Genero reporting application to the Genero Report Engine. It is used by the report design document, giving the designer the ability to select fields for placement on a report and setting report triggers. See [Generate a data schema from a Genero BDL report program](#) on page 572.

5. Create a report design document.

A report design document (4rp) is a file that defines a single report. For each report you need, you would create a separate report design document. When it comes time to create a specific report, you tell the report application which file to use.

- a) Select **File >> New, Reports** and select either **Empty Report (.4rp)** or **List Report (.4rp)** to start a new report.
- b) On the Data View tab, select the data schema.
- c) Report output can be output in various formats. Select **File >> Report properties >> Output Configuration** to change the default output for the report.
- d) Save the report design document and add it to the **Designs** virtual folder in your application node.

For more information on working with the report design document, see [The Report Design Document](#) on page 657.

6. Build the report application.

Right-click the report application node or the report group node and select **Build** to compile and link the application.

Run the Report (GRW for BDL)

- Right-click the report application node in the **Projects** view and select **Execute** to run the program.
 - Note:** If the report data will come directly from a database, your report program must have access at execution to the database where the data is stored.
- By default a new report is in SVG format, and the output is set to Preview using the [Genero Report Viewer](#); you can change these defaults in your Report Design document, or in your report program code.

BDL programs

You can code a reporting application using Genero Business Development Language.

- [Writing the Genero BDL report program](#) on page 564
- [Create labels: the report program \(Genero BDL\)](#) on page 580
- [Report Data Wizard](#) on page 581

Writing the Genero BDL report program

A Genero BDL report program retrieves the data and outputs it to a report.

- [Overview](#) on page 564
- [Tips](#) on page 565
- [Creating a simple report](#) on page 565
- [Fetching report data](#) on page 569
- [Output options](#) on page 571
- [Generate a data schema from a Genero BDL report program](#) on page 572
- [Allowing the user to select output options](#) on page 572
- [Running a Genero ASCII report using GRW \(Compatibility Report\)](#) on page 574
- [Sub reports](#) on page 575

Overview

Genero BDL is used to write the code for a 4gl file that retrieves the data, often from a database, and outputs it to a report.

This BDL program contains:

- The [Report Driver](#), which specifies the data the report will include. If you are retrieving data from database tables, for example, a database cursor is defined to retrieve specific rows. The Report Driver stores the database values in program variables, and sends these - one record at a time - to the REPORT program block.
- The [REPORT program block](#), which specifies the control blocks for a report, and sends the data to the Genero Report Engine. Variables can be defined to allow calculations on the data values, which are output along with the data values.

The Genero Report Engine uses the data and the Report Design document (4rp - created using Genero Report Designer) to process and output the report, in accordance with the [reporting API functions](#) that have been called by the BDL program.

See [Steps to a Report \(GRW for BDL\)](#) on page 562 for a complete outline of the reporting process.

[Some Tips for Legacy Report Designers](#) provides information about the correlation between Report Designer and traditional 4GL commands in reports.

The `OrderReport.4gl` example program included in the GRW demo **Reports**:

- Retrieves the data and outputs it to the Report Engine.
- Allows the user to [select the desired report](#) and the output settings, by displaying a form (`Configuration.42f`) for input.
- Uses the Reporting API function `fgl_report_loadCurrentSettings()` to load the original settings from the 4rp file. (Additional [optional functions](#) can specify the changes to be made to those settings, based on the user input.)

- Uses the Reporting API function `fgl_report_commitCurrentSettings()` to commit the changes this program has made.
- The Form Design file for this form, (`Configuration.4fd`) is one of the files in the GRW demo project, **Reports**.

See the [Reporting API](#) for the specific functions that you can use to control the output of a report; some examples are in [Change Report Output Options](#).

Tips

Tips for writing a BDL report program.

1. The [Reporting API](#) functions `fgl_report_loadCurrentSettings` and `fgl_report_CommitCurrentSettings` are required and must be called in sequence. (Calls to these functions bracket any calls to other functions that change the default output options.) These functions must be called in the program prior to executing the START REPORT command in the Report Driver. If your program does not change the default output options, you can simply call the single function `fgl_report_loadAndCommit` instead.
2. Always use ORDER EXTERNAL in the REPORT program block to tell the report that the input records have already been sorted by the SELECT statement, and there is no need for the program to resort the fields. This will improve performance.
3. Restrict your use of control blocks in the REPORT program block to FIRST PAGE HEADER, BEFORE GROUP OF, AFTER GROUP OF, ON EVERY ROW, or ON LAST ROW. See [The Report Program Block](#).
4. Prefer the use of fields over expressions to hold report values. Since expressions don't have names, it will be difficult to reference them in the [Report Design Document](#) (they could only be referenced by index).
5. Prefer calculating values in the **REPORT** program block over calculating in the Report Design Document, to use the power of the DVM.
6. Don't be stingy when outputting fields; this will enhance the ability to use the BDL source for multiple report formats. Define a record, **orderline** for example, that contains all the data fields retrieved from the database by the database cursor; use that record definition in the OUTPUT to REPORT statement:

```
OUTPUT TO REPORT <reportname> (orderline.*)
```

Creating a simple report

This example BDL program (`SimpleReport.4gl`) uses data from the officestore database to create a report. There are more complex examples in the Reports project provided as a demo with Genero Report Writer.

Type Definition

This section gives the database schema used in variable definitions, and defines a User Type that consists of a single record containing all the fields from all the referenced tables.

Creating a TYPE allows the record definition to be specified only once in the program; thereafter, the name of the TYPE is used in the program wherever that record definition would be required.

```
SCHEMA officestore

TYPE ORDERTYPE RECORD
    orders      RECORD LIKE orders.* ,
    account     RECORD LIKE account.* ,
    country     RECORD LIKE country.* ,
    lineitem    RECORD LIKE lineitem.* ,
    product     RECORD LIKE product.* ,
    category    RECORD LIKE category.* ,
    item        RECORD LIKE item.*
END RECORD
```

MAIN program block

The **MAIN** program block contains the program logic that allows a user to run a report.

- defines a report handler object.
- call the mandatory BDL functions that configure the report:
 - `fgl_report_loadCurrentSettings` accepts the name of the Report Design document (`4rp`) as a parameter. The return value is a boolean indicating whether the load of the settings from the `4rp` file was successful.
 - `fgl_report_commitCurrentSettings` returns a `SaxDocumentHandler` object, defined as **handler**.
- calls the **Report Driver function**, `runReportFromDatabase`, to run the report, passing **handler**.

Note: The example `runReportFromDatabase` function contains the START REPORT statement. Do not place any code between the call to `fgl_report_commitCurrentSettings` and the START REPORT statement that would allow the user to cancel the report

```

MAIN

  DEFINE handler om.SaxDocumentHandler-- report handler

  --call the mandatory functions that configure the report
  IF fgl_report_loadCurrentSettings("myreport.4rp") THEN -- if the file
                                                    -- loaded OK
    LET handler = fgl_report_commitCurrentSettings() -- commit the
file                                                    -- settings
  ELSE
    EXIT PROGRAM
  END IF

  -- run the report by calling the report driver contained
  -- in your function runReportFromDatabase
  IF handler IS NOT NULL THEN
    CALL runReportFromDatabase(handler)
  END IF

END MAIN

```

Important: The `libgre.42x` library contains these mandatory functions and other BDL helper functions for Reports. This library must be included in any report application. List this library in the **external dependencies property** of any Genero Studio Project Manager application node that uses Genero Report Writer.

The Report Driver

The `runReportFromDatabase` function uses SQL to extract the data from the database officestore:

- **handler** previously created by the `fgl_report_commitCurrentSettings` function is passed to this function
- defines a record variable **orderline** using the User Type defined in MAIN
- defines **handler** passed as a parameter to this function
- makes a connection to the database in unique-session mode, since the program will not need to connect to other databases.
- declare a cursor for the SQL statement.
- uses the **ORDER BY** clause of the SQL statement defines the sort order of the data.
- These next statements are only used when the data is being retrieved from a database.
 - the BDL statement **START REPORT <reportname> TO XML HANDLER** must be used to instantiate the report driver, using the **handler** that was passed to this function; this specifies that the report data should be output in XML format.

- the **FOREACH** loop opens the cursor and fetches the data from the database into the record variable, one row at a time.
- the **OUTPUT TO REPORT** statement outputs each data row to the REPORT program block.
- the **FINISH REPORT** terminates the BDL report process.
- closes the SQL cursor.

```

FUNCTION runReportFromDatabase(handler)
  DEFINE orderline ORDERTYPE,           -- User Type defines record
        handler om.SaxDocumentHandler -- definition for parameter
                                          -- passed to this function

  DATABASE "officestore"  -- database connection
  DECLARE c_order CURSOR FOR  -- cursor declaration
  SELECT orders.*,
        account.*,
        country.*,
        lineitem.*,
        product.*,
        category.*,
        item.*
  FROM orders, account, lineitem, product, category, item, country
  WHERE
    orders.orderid = lineitem.orderid
    AND orders.userid = account.userid
    AND lineitem.itemid = item.itemid
    AND item.productid = product.productid
    AND product.catid = category.catid
    AND country.code = orders.billcountry
  ORDER BY orders.userid, orders.orderid, lineitem.linenum

  START REPORT report_all_orders TO XML HANDLER handler -- handler that
was                                                    -- passed to this
function
  FOREACH c_order INTO orderline.*  -- use cursor to fetch
data
  OUTPUT TO REPORT report_all_orders(orderline.*) -- send data to
report
                                                    -- function
  END FOREACH
  FINISH REPORT report_all_orders
  CLOSE c_order

END FUNCTION

```

See the *Genero Business Development Language User Guide* for additional information about the use of cursors, connections, and BDL report statements.

The REPORT program block

This program block accepts the data from the driver, specifies the order in which the data was sorted, and outputs the data to be formatted as specified in the report design page (4rp).

The **FORMAT** section specifies the control blocks for a report. The use of each control break is optional, depending on the requirements of your report document. We recommend that you restrict your usage to these control blocks:

- **FIRST PAGE HEADER** - specifies the action that the runtime system takes before it begins processing the first input record.

- BEFORE GROUP OF/AFTER GROUP OF - specifies the action the runtime system takes before or after it processes a group of input records.
- ON EVERY ROW - specifies the action the runtime system takes for every input record that is passed to the report definition.
- ON LAST ROW - specifies the action the runtime system takes after it processes the last input record that was passed to the report definition and encounters the `FINISH REPORT` statement.

See BDL Reports in the *Genero Business Development Language User Guide* for a complete discussion of the BDL statements associated with a **REPORT** block.

Note: Since PAGE HEADER and PAGE TRAILER are triggered based on the line count of the BDL report, which does not correspond with the actual page breaks, their usage should be avoided. Create [page headers and footers](#) in the report design document instead.

Example REPORT block from `SimpleReport.4gl`:

- define variables to allow calculations on the data values; the calculations are output along with the data.
- the **ORDER EXTERNAL BY** statement informs the report that the data was retrieved, and will be output, in the specified sort order
- the **FORMAT** section of the report uses control blocks to set the values of the report variables used to store calculations and to send the report data and calculations to the report. Unlike a Genero report, the **PRINT** statement does not contain any formatting of the data or the report line (SPACES, LINE_NO, and COLUMN operators aren't used, for example.) The report format is specified in the Report Design document (4rp).
- set the variable **overalltotal** to zero at the control break at the beginning of the report (**FIRST PAGE HEADER**).
- re-set the variable **ordertotal** to zero each time the value of orderid changes in the data received from the report driver (**BEFORE GROUP OF**).
- For each row of data received from the report driver (**ON EVERY ROW**), these lines store some calculations in variables. The **PRINT** statement outputs the database data and variables to the report, in XML format.

```
REPORT report_all_orders( orderline )
  DEFINE
    orderline      ORDERTYPE,
    lineitemprice  LIKE lineitem.unitprice,    -- total price for item
    overalltotal   LIKE orders.totalprice,     -- accumulator for total
price
    ordertotal     LIKE orders.totalprice      -- accumulator for total
price
                                                    -- for order
                                                    -- for report

  -- specify the order of the sorted data resulting from SQL statement
  ORDER EXTERNAL BY orderline.orders.userid, orderline.orders.orderid,
                    orderline.lineitem.linenum

  FORMAT
  FIRST PAGE HEADER
    LET overalltotal=0                                -- initialize report total

  BEFORE GROUP OF orderline.orders.orderid
    LET ordertotal=0                                  -- initialize ordertotal for
                                                    -- each new order

  -- after calculations for each data row, output the data and
  -- the calculations to the Report Engine
  ON EVERY ROW
    LET lineitemprice = orderline.lineitem.unitprice *
                    orderline.lineitem.quantity
    LET overalltotal=overalltotal + lineitemprice
```



```

LET ordertotal=ordertotal + lineitemprice
PRINT orderline.*, lineitemprice, overalltotal, ordertotal

END REPORT

```

Fetching report data

You can obtain the data for the report in several ways.

From a database using SQL

You can use SQL statements to retrieve the data for the report from database tables, requiring a connection to the database and the use of an SQL cursor.

The `SimpleReport.4gl` program uses this technique, shown in the [runReportFromDatabase](#) function.

From a data file

You can get the report data from a data file - one created by the BDL UNLOAD statement, for example.

This example uses the **OrderReport.unl** file in the GRW demo Reports.

The only change to the MAIN program block would be the call to the [runReportFromFile](#) function in line 26.

```

MAIN

  DEFINE handler om.SaxDocumentHandler -- report handler

  --call the mandatory functions that configure the report
  IF fgl_report_loadCurrentSettings("myreport.4rp") THEN -- if the file
                                                    -- loaded OK
    LET handler = fgl_report_commitCurrentSettings() -- commit the
file                                                    -- settings
  ELSE
    EXIT PROGRAM
  END IF

  -- run the report by calling the report driver contained in your
  -- function runReportFromFile
  IF handler IS NOT NULL THEN
    CALL runReportFromFile(handler)
  END IF

END MAIN

```

The function **runReportFromFile** replaces the **runReportFromDatabase** function as the Report Driver. It uses the unload file **OrderReport.unl** to provide the data for the report.

- Defines **ch** as a variable of type **base.channel**. The BDL Channel class provides read/write access to files.
- Defines the variable **dataFile** to specify the unload file containing the data.
- Assigns the report name to the **dataFile** variable.
- Creates the **ch** channel object.
- opens the channel to the data file **OrderReport.unl** in read (r) mode.
- The WHILE statement reads a line from the data file, providing the variable list enclosed in brackets. The line is output to the [REPORT program block](#). The statement terminates when all the lines have been read.
- Closes the channel.

```

FUNCTION runReportFromFile(handler)
  DEFINE
    orderline OrderType,
    handler om.SaxDocumentHandler,
    ch base.channel, -- definition of channel object

```

```

        dataFile String          -- file containing report data

    LET dataFile = "./OrderReport.unl"
    LET ch = base.Channel.create()
    CALL ch.openFile(dataFile,"r")

    START REPORT report_all_orders TO XML HANDLER handler
    WHILE ch.read([orderline.*])
        OUTPUT TO REPORT report_all_orders(orderline.*)
    END WHILE
    FINISH REPORT report_all_orders

    CALL ch.close()
END FUNCTION

```

From an XML file

You can get the report data from an XML file instead of getting it directly from a database.

The XML file can be a ProcessLevelData file created by a Genero report ([Output to an XML File](#)) which has a special format, or it may be an arbitrary XML file.

From A ProcessLevelData file

This XML file has a special format, and is created using the Report Writer API function [fgl_report_createProcessLevelData File\(\)](#).

The MAIN block of SimpleReport.4gl would change as follows:

- Provides the name of the data file to be used for the report
- Use the BDL function [fgl_report_runReportFromProcessLevelDataFile](#) to run a report using a data file. The parameters are:
 - The report handler object
 - A String containing the name of the XML file that contains the data
- This function will replay the report from the file thereby replacing the running of the report (START REPORT, OUTPUT TO REPORT, FINISH REPORT statements).

```

MAIN

    DEFINE handler om.SaxDocumentHandler, -- report handler
           data_file String,             -- XML file containing data
           report_ok boolean              -- return value from function

    --call the mandatory functions that configure the report
    IF fgl_report_loadCurrentSettings("myreport.4rp") THEN -- if the file
                                                         -- loaded OK
        LET handler = fgl_report_commitCurrentSettings() -- commit the file
                                                         -- settings
    END IF

    --call the report driver to run the report
    IF handler IS NOT NULL THEN
        LET data_file = "./OrderReportData.xml"
        LET report_ok =
            fgl_report_RunReportFromProcessLevelDataFile(handler, data_file)
    ELSE
        EXIT PROGRAM
    END IF

END MAIN

```

From Other XML files

XML files that were not created by the Reporting API function [fgl_report_createProcessLevelData File\(\)](#) can be used as the data source for a Genero report. The file `OrderData.xml` in the [demo sample files](#) of the Reports project is an example of this type of file.

The function [fgl_report_runFromXML](#) is used to specify the data source:

- This function sets up the report engine based on the current settings that have previously been loaded by a call to [fgl_report_loadCurrentSettings\(\)](#), and may have been modified by calls to [fgl_report_selectDevice\(\)](#) or [fgl_report_selectPreview\(\)](#). The function automatically calls [fgl_report_commitCurrentSettings\(\)](#).
- This function will replay the report from the file thereby replacing the running of the report (START REPORT, OUTPUT TO REPORT, FINISH REPORT statements).

This example is from the `OrderReportXML.4gl` demo program in the Reports project:

```

MAIN

  IF NOT fgl_report_loadCurrentSettings("Table.4rp") THEN
    EXIT PROGRAM
  END IF

  IF NOT fgl_report_runFromXML("OrderData.xml") THEN
    DISPLAY "RUN FAILED"
    EXIT PROGRAM
  END IF

END MAIN

```

From a Web Service

Web Services can be used to fetch data for a report. BDL functions allow you to access a web service and retrieve data, storing it in BDL program variables.

See **Using Web Services** in the *Genero Studio User Guide*, and the *GWS User Guide* for additional information and examples.

Output options

Various output formats are available for a report, to include SVG, PDF and image files. The output will be sent to the destination specified by the output options.

Default Output

If you don't override the default options, on report execution the user will:

- Create an SVG report
- Preview the Report in Genero Report Viewer (for desktop reporting applications) or Genero Report Viewer for HTML5 (for web-based reporting applications).

From the Genero Report Viewer, a user can then direct the output to a printer.

The Reporting API provides BDL functions to change the output in your BDL program by overriding the default options. Some examples are provided in [Changing Output Options](#); also see [Allow User to Select Output](#).

Output Options in the 4rp (Report Design Document)

You can define paper settings and output options for a report within the report design document itself. To do this, open a report in the Genero Report Designer, and select **File >> Report properties >> Paper settings** or **File >> Report properties >> Output options**. These values can be changed at runtime, using the reporting APIs. See [Change paper settings and output format](#) on page 589.

See also: [Configuring Fonts and Printers](#)

Generate a data schema from a Genero BDL report program

After you write or modify a Genero report program, you must generate the data schema (`rdd`) file. This file is used by the Genero Report Designer to provide a list of data objects for use in the report design.

The data schema (`rdd`) file is based on the SQL statement in your Genero report application source file (`4gl`). This `rdd` file is used in the report design document (`4rd`) to populate the [Data View](#), providing details about the fields that will be streamed by the application. The schema contains the list of database columns that make up your data record, as well as grouping details.

Although the data for the report originally may have come from several different data tables, the PRINT statement in your BDL REPORT program block outputs the data as part of a single record. See the Genero Studio >> Report Writer documentation topic "Writing the BDL Program" for more information.

From the command line

Use the `--buildrdd` command-line option of the `fglcomp` tool to create a data schema (`rdd` file). For example:

```
fglcomp --build-rdd SimpleReport.4gl
```

The output of this command will be `SimpleReport.rdd`. The `rdd` file will be stored in the same location as the `4gl` file.

Note: If your Genero program contains multiple `4gl` files, run `fglcomp` against the file containing your REPORT program block.

From Genero Studio

Add `--build-rdd` to the **Compiler options** property for your Genero source (`4gl`) file to generate the `rdd` file automatically each time the `4gl` file is compiled. Select the `4gl` file listing in the application node of the Project tab to display its properties in the Properties View. The `rdd` file will be stored in the directory specified in the **Target Directory** property of the application node that contains the `4gl` file.

Allowing the user to select output options

Users can choose the format to see the report output.

In the GRW demo program **Order Report**, this form allows the user to make choices regarding the output:

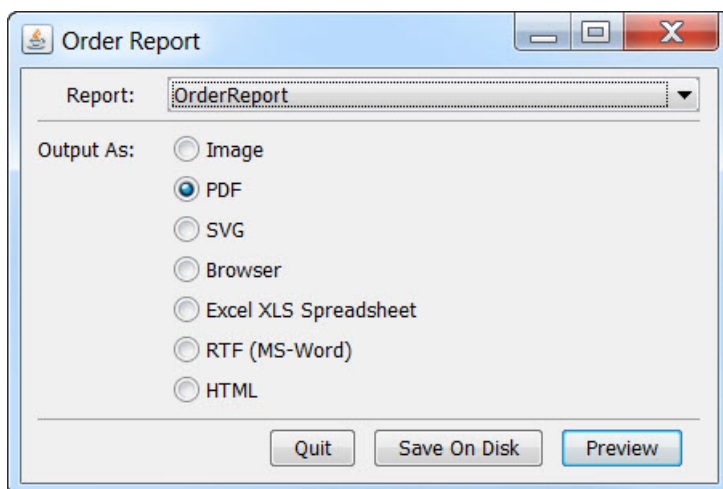


Figure 314: Report demo Form

This code implements this functionality. The form definition file, `Configuration.4fd`, is included in the demo files for **Reports**, the [GRW Demo](#).

The MAIN block of `SimpleReport.4gl` would change as follows:

- Call a function named `select_output` that displays the form and returns the user's choices

- Configure the report using the mandatory Reporting API functions plus these additional functions:
 - [fgl_report_selectDevice](#) - sets the report output device; the parameter is one of: "Printer", "PDF", "Image", "SVG".
 - [fgl_report_selectPreview](#) - sets the preview option; the parameter is TRUE (preview) or FALSE (no preview).

```

MAIN

  DEFINE handler om.SaxDocumentHandler, -- report handler
         data_file String,             -- XML file containing data
         report_ok boolean,            -- return value from function
         r_filename STRING,            -- filename of desired report
         r_output STRING,              -- output format
         preview INTEGER               -- preview indicator

  -- call the mandatory functions that configure the report using the
  -- choices made by the user
  -- in your new select_output function
  CALL select_output() RETURNING r_filename, r_output, preview
  IF fgl_report_loadCurrentSettings(r_filename) THEN -- if the file loaded
OK
    CALL fgl_report_selectDevice(r_output)           -- changing default
    CALL fgl_report_selectPreview(preview)          -- changing default
    LET handler = fgl_report_commitCurrentSettings() -- commit changes
  ELSE
    EXIT PROGRAM
  END IF

  -- run the report by calling the report driver contained in
  -- your function runReportFromDatabase
  IF handler IS NOT NULL THEN
    CALL runReportFromDatabase()
  END IF

END MAIN

```

select_output - this new function displays a form allowing the user to specify the name of the report to be run (the name of the 4rp file), the output format, and whether a preview is preferred.

```

FUNCTION select_output()
  DEFINE
    r_output STRING, -- output format option
    r_action STRING, -- used to set preview option
    r_filename STRING, -- filename of Report Design document (4rp
filename)
    preview INTEGER -- preview option TRUE/FALSE, to display
-- report in Report Viewer

  CLOSE WINDOW SCREEN
  OPTIONS INPUT WRAP
  -- display form allowing user to select the 4rp filename and output
options
  OPEN WINDOW f_configuration WITH FORM "Configuration"
  LET INT_FLAG=FALSE
  LET preview = FALSE
  INPUT BY NAME r_filename,r_output,r_action
    ON ACTION CANCEL
      EXIT PROGRAM
    ON ACTION CLOSE
      EXIT PROGRAM
  END INPUT

```

```

CLOSE WINDOW f_configuration
CALL ui.interface.refresh()

-- set preview variable to match output option r_action
IF r_action IS NOT NULL THEN
  IF r_action == "preview" THEN
    LET preview = TRUE
  END IF
END IF

-- return report name (filename of Report Design Document) and output
-- option variables to be used to configure the report engine
RETURN r_filename || "4rp", r_output, preview

END FUNCTION

```

Running a Genero ASCII report using GRW (Compatibility Report)

This type of report is referred to as a compatibility report.

If you have an existing Genero BDL program (4g1) that creates an ASCII report, you can execute the program using the Genero Report Writer without changing the existing Report Driver or the existing Report program block, and without creating a [Report Design document](#) (4rp).

These additional lines would be made in the MAIN program block of the 4g1 file, to call the mandatory API functions that configure the report:

```

DEFINE handler om.SaxDocumentHandler

IF fgl_report_loadCurrentSettings(NULL) THEN           -- switch on
  Compatibility mode                                 -- (run without 4rp file)
  LET handler = fgl_report_commitCurrentSettings() -- commit settings
END IF

```

Switching on Compatibility mode overrides the "TO ..." part of the START REPORT BDL statement.

The formatting of the report will be that specified in the REPORT program block of your BDL program. (See [Auto-formatting](#) for an alternative to the Compatibility format.) The report engine will use the default output settings, displaying a preview of the report in SVG format in the Report Viewer. Additional SVG preview options are available, using the API function [fgl_report_configureSVGPreview](#). The API function [fgl_report_configureCompatibilityOutput](#) can optionally be used to change the default output settings.

The demo report **OrderReportASCII** illustrates this feature.

Auto-formatting Reports that have no 4rp (Report Design Document)

Beginning with version 2.40, auto-formatting of these reports that have no report design document (4rp) is available, as an alternative to the [Compatibility format](#). New API calls have been added to provide generic report formatting. Currently only one design is provided, a simple list design that is compatible with the List Report template that is shipped with Genero Report Writer. This type of auto-formatting is particularly well suited to produce Excel output from arbitrary reports. See [fgl_report_setAutoformatType](#) and [fgl_report_configureAutoformatOutput](#).

Report Metadata in Compatibility Reports

New API functions allow you to add report metadata to Compatibility Reports. These functions are intended to be used for compatibility reports only. See [Report Metadata functions](#). For regular reports, the metadata can be set in corresponding properties in the Properties View of the Report Designer.

Mixing Genero ASCII Reports, GRW Reports, and Compatibility Reports

Genero ASCII (text-based) reports and GRW (graphical) reports can be run from the same Genero program without requiring any additional calls.

However, running [Compatibility Reports](#) requires that Compatibility mode be switched on. Once Compatibility mode is invoked, it stays active throughout the program, running any subsequent Genero ASCII reports using the Genero Report Writer also. If your program needs to restore text-based output for a Genero ASCII report, you must call the API function [fgl_report_stopGraphicalCompatibilityMode](#).

Sub reports

Sub reports allow one report to be called from another.

- [What are sub reports?](#) on page 575
- [Use cases for sub reports](#) on page 575
- [Creating a master report data source](#) on page 576
- [Creating a master report design](#) on page 579

What are sub reports?

Sub reports allow one report to be called from another.

Sub reports allow reuse of BDL code in the report application and reuse of available report designs. Sub reports can be used to break complex designs into smaller reusable parts.

In the BDL code, a sub report call is detected when a START REPORT instruction is found within another report. The report containing the START REPORT instruction is called the **master report**, and the called report is the **sub report**.

Runtime behavior

At runtime, the reports are combined. The sub report is inserted in the master report.

The behavior at the insertion point follows the same rules that generally apply to nested containers. The master report needs to be designed with awareness of the configuration of the root container of the sub report. For example, if the sub report occupies all the space it can get (the height and width of the MiniPage both set to max), the master report must be prepared to handle that. In particular, the use of ancestors other than MiniPage should be avoided, since they will become overfull if the sub report requires more than one page of space.

Use cases for sub reports

Create a report composed of other reports

You have two reports that you want to output as a single report. For example, you have a report that is a graph showing revenue by region, and you have a list report providing the details on the revenue by region. You can create a master report that outputs both of these reports - the graph report and the line detail report- as a single report.

Reuse a report part across multiple reports

Consider an order confirmation report and an invoice report. The two reports could share a table containing the sales items and their description (the shared part). You can put that table in a sub report. Changing the sub report then changes the two reports using the sub report.

Create a report where a sublist has more than one sort key item

Imagine you have a report with two loops, where each loop defines a sublist:

```
REPORT report(r)
. . .
FORMAT
ON EVERY ROW
```

```

PRINT r.*
#loop1
  DECLARE c1 CURSOR FOR
    SELECT * FROM t1 WHERE t1.key=r.t1key ORDER BY a,b,c
  FOREACH c1 INTO t1.*
    PRINT t1.*
  END FOREACH
#loop2
  DECLARE c2 CURSOR FOR
    SELECT * FROM t2 WHERE t2.key=r.t2key ORDER BY id
  FOREACH c2 INTO t2.*
    PRINT t2.*
  END FOREACH
END REPORT

```

Should the loops loop1 and loop2 be replaced by sub reports?

Loop1 specifies three sort key items in the `ORDER BY` clause. Replacing loop1 with a sub report using `ORDER EXTERNAL a, b, c` gives you the flexibility to "trigger" on a, b, and c in the design.

Replacing loop2 by a sub report would not provide any gain (assuming there is no interest to group on `id`). For this loop, you might prefer to see the whole design.

Do NOT use sub reports to process nested sub lists

Genero reports support arbitrarily complex models using `PRINT` statements inside iterator statements (`WHILE`, `FOR`, `FOR EACH`) and conditional statements (`IF`, `CASE`). It is not necessary to use sub reports for this purpose. When sub reports are used for this purpose to produce complex reports (such as an invoice report), the report becomes scattered across numerous BDL REPORTs and report design (4rp) files, obscuring the overall structure. The impossibility to see the design as a whole must be taken into consideration when using this strategy.

Creating a master report data source

A sub report is a report that is started from within another report via a `START REPORT` statement.

The `START REPORT` statement for the sub report takes no arguments. Do not add `TO XML HANDLER` for the sub report call.

Sub reports can be called from any control block wherever printing in loops is allowed: `ON EVERY ROW`, `BEFORE GROUP`, `AFTER GROUP`, or `ON LAST ROW`.

The `START REPORT` must be in the calling report block. The `START REPORT` statement cannot be placed inside a function called from the report.

The `START REPORT`, `OUTPUT TO REPORT`, and `FINISH REPORT` statements must be in the same control block. The `OUTPUT TO REPORT` and `FINISH REPORT` statements can be placed inside a function or functions called from the report, however those functions must be in the same control block as the `START REPORT` statement.

You can have multiple calls to sub reports from the same control block, however the calls cannot be nested.

Valid:

```

REPORT master_report(...)
...
FORMAT
  BEFORE GROUP ...
  ...
  START REPORT detail_report_1
  FOREACH ...
    OUTPUT TO REPORT detail_report_1(...)
  END FOREACH
  FINISH REPORT detail_report_1

```



```

START REPORT detail_report_2
FOREACH ...
  OUTPUT TO REPORT detail_report_2(...)
END FOREACH
FINISH REPORT detail_report_2
...
END REPORT

```

Not valid:

```

REPORT master_report(...)
...
FORMAT
  BEFORE GROUP ...
  ...
  START REPORT detail_report_1
  START REPORT detail_report_2 -- Not allowed until after
                                -- FINISH REPORT detail_report_1
  ...
END REPORT

```

Sub reports can be nested to arbitrary depth, but recursion is not allowed.

```

REPORT master_report(...)
...
FORMAT
  ON EVERY ROW
  ...
  START REPORT detail_report
  FOREACH ...
    OUTPUT TO REPORT detail_report(...)
  END FOREACH
  FINISH REPORT detail_report
END REPORT

REPORT detail_report(...)
...
FORMAT
  ON EVERY ROW
  ...
  START REPORT other_detail_report -- Since recursion is not allowed, we
are                                     -- not allowed to call "master-
report" or                               -- "detail_report" from here.
  FOREACH ...
    OUTPUT TO REPORT other_detail_report(...)
  END FOREACH
  FINISH REPORT other_detail_report
END REPORT

```

Detected sub report calls are shown in the report designer. They appear as nodes in the data view, triggers the structure view, and images in the document view.

Code Example

This code example shows a master report **master_report** that calls two sub reports (**report_orders** and **report_items**) from the ON EVERY ROW section. The invocation of a sub report requires the START REPORT, OUTPUT TO REPORT and FINISH REPORT from within the master report.

The `START REPORT` instructions for sub reports cannot include the `TO` clause (`START REPORT repname TO XML HANDLER`); doing so will yield unexpected results.

```

FUNCTION run_master_report
...
  START REPORT master_report TO XML HANDLER handler
  FOREACH ...
    OUTPUT TO REPORT master_report
  END FOREACH
  FINISH REPORT master_report
END FUNCTION

REPORT master_report()
  DEFINE
    orderline OrderType,
    ch base.channel

  FORMAT
    ON EVERY ROW

    # Data source for first sub report
    LET ch = base.Channel.create()
    CALL ch.openFile(dataFile,"r")

    # First sub report
    START REPORT report_orders -- Notice there is no TO XML HANDLER
                                -- for the sub report

    WHILE ch.read([orderline.*])
      OUTPUT TO REPORT report_orders(orderline.*)
      IF fgl_report_getErrorStatus() THEN
        DISPLAY "FGL: STOPPING REPORT, msg=
\"",fgl_report_getErrorString(),\""
        EXIT WHILE
      END IF
    END WHILE
    FINISH REPORT report_orders

    CALL ch.close()

    # Data source for second sub report
    LET ch = base.Channel.create()
    CALL ch.openFile(dataFile,"r")

    # Second sub report
    START REPORT report_items -- Notice there is no TO XML HANDLER
                               -- for the sub report

    WHILE ch.read([orderline.*])
      OUTPUT TO REPORT report_items(orderline.*)
      IF fgl_report_getErrorStatus() THEN
        DISPLAY "FGL: STOPPING REPORT, msg=
\"",fgl_report_getErrorString(),\""
        EXIT WHILE
      END IF
    END WHILE
    FINISH REPORT report_items

    CALL ch.close()

  END REPORT

  REPORT report_orders
  ...
END REPORT

```

```
REPORT report_items
...
END REPORT
```

Creating a master report design

Create a new empty report design document (4rp). Select the master report data source as the data schema. Detected sub report calls are shown in the report designer. They appear as nodes in the data view, triggers the structure view, and images in the document view.

The Data View

In the data view, the invocation of the sub report is displayed at its position in the application source. While the execution of the sub report requires three invocations (START REPORT, OUTPUT TO REPORT and FINISH REPORT), only the location of the START REPORT item is shown in the data view.

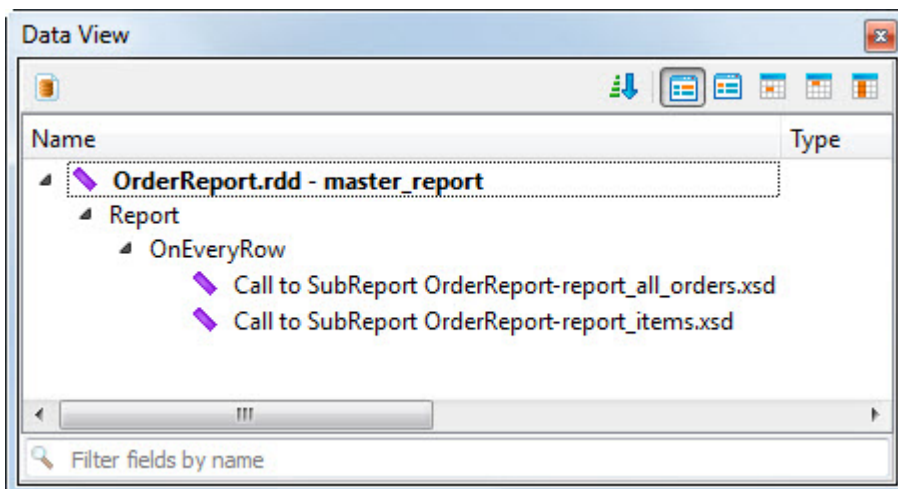


Figure 315: Sub reports in the Data View

The Report Structure view

A sub report trigger is created in the document report structure. Set the URL property of the sub report trigger to a report design that matches the sub report schema.

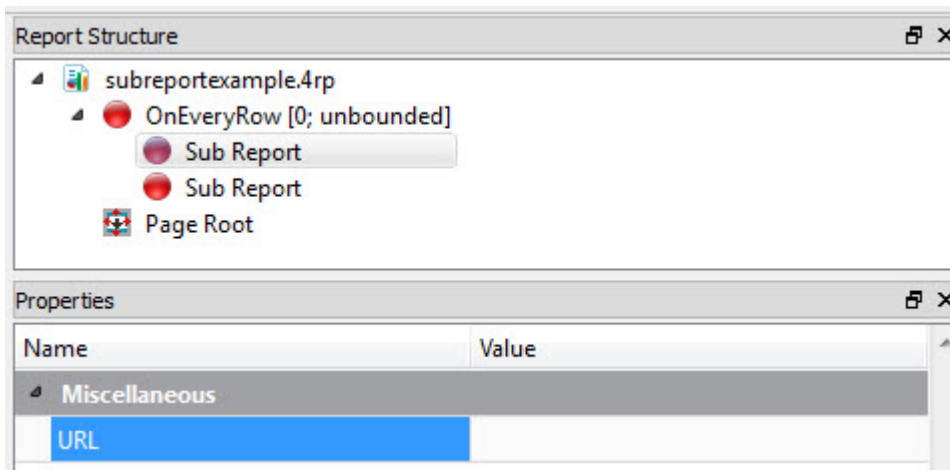


Figure 316: Sub report triggers in the Report Structure

Document view (Central Work Area)

Sub reports display as images in the document view in the Central Work Area. Double-click the image to open the design file for editing.

Create labels: the report program (Genero BDL)

A common use of reports is to create a page of labels, such as address labels. Creating labels requires you to create a report design document representing a single label and a report application that prints multiple labels on a page.

Create the report design document

The report design document (.4rp) contains the design for a single label. You can put anything you wish on the label, as long as it fits onto the page.

See [Design labels](#) on page 684 for details on creating the single-label report design document (.4rp).

Update your report application

Once the label is defined in a report design document, you update your report application so that it outputs the data needed for a label within `ON EVERY ROW` in the `REPORT` block, and it tells the report engine to output multiple labels onto a single physical page.

- In the `FORMAT` section of the `REPORT` program block, output the data needed for the label `ON EVERY ROW`.
- Configure the report engine for labels using Reporting API functions.
 - `fgl_report_selectLogicalPageMapping` allows you to specify that this report will create a page of labels.
 - `fgl_report_setPaperMargins` sets the margins (top, bottom, left, right).
 - `fgl_report_configureLabelOutput` configures the physical layout of the label page using these parameters:
 - paper Width
 - paper Height
 - label Width (set to null)
 - label Height (set to null)
 - number of labels per row
 - number of labels per column

Example

This example uses the `OrderLabels.4rp` report design document from the Reports GRW demo.

In the [SimpleReport BDL program](#), the `MAIN` program block that configures the report engine is changed as follows:

- Loads `OrderLabels.4rp` as the [Report Design Document](#)
- The function `fgl_report_selectLogicalPageMapping` allows you to specify the format of the page as "labels"
- The function `fgl_report_setPaperMargins` sets margins using these parameters:
 1. top margin
 2. bottom margin
 3. left margin
 4. right margin
- Function `fgl_report_configureLabelOutput` - configures the physical layout of a label page using these parameters:
 1. paper Width

2. paper Height
3. label Width
4. label Height
5. number of labels per row
6. number of labels per column

```

-- configure report engine; the functions prefixed fgl that are called
here
-- are part of the Reporting API
IF fgl_report_loadCurrentSettings("OrderLabels.4rp") THEN -- load the
labels
-- design document (4rp
file)
-- special settings for labels
CALL fgl_report_selectLogicalPageMapping("labels")
CALL fgl_report_setPaperMargins("5mm", "5mm", "4mm", "4mm")
CALL fgl_report_configureLabelOutput("a4width", "a4length", null, null, 2, 6)
END IF
LET handler = fgl_report_commitCurrentSettings() -- commit changes

```

The rest of the [SimpleReport program](#) would remain unchanged. This example would print two labels across and six labels down per page.



Figure 317: Output of labels on a physical page

Report Data Wizard

- [Using the Report Data Wizard](#) on page 581
- [Example Wizard code](#) on page 586
- [Using the example code](#) on page 587

Using the Report Data Wizard

A Wizard is now available to generate BDL code for your Genero report program (4g1). Select **File>>New, Reports, Report from Database (.4gl)** to display the dialogs that guide you in creating the SQL query statement that will extract the report data from your database.

Table Selection Page

Select the database and tables for the query.

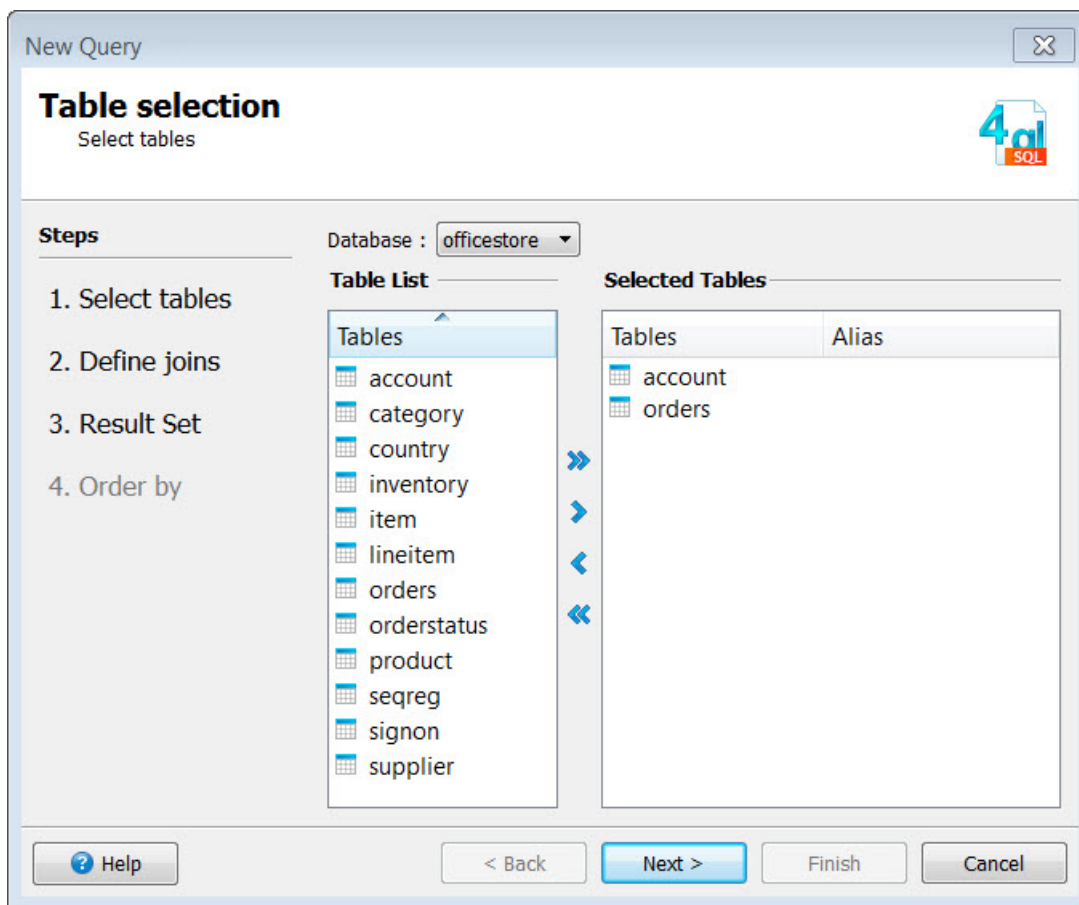


Figure 318: New Query wizard, Table Section page

Press **Next** to continue.

Define Joins Page

The **New Query** dialog is used to add the joins between the tables to be used in the SQL query. The information is pre-filled from the database meta-data in the database meta-schema file (4db). You can select different columns in the tables, or select a different join method.

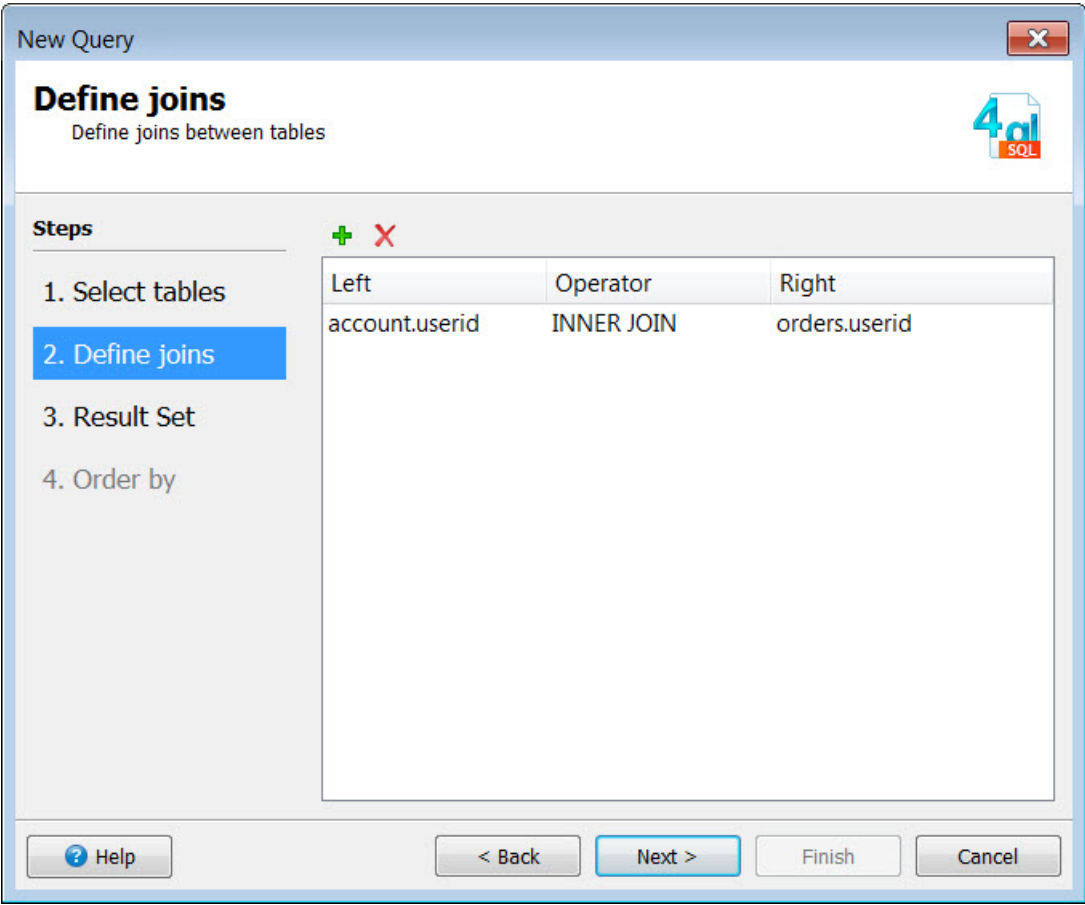


Figure 319: New Query wizard, Define joins page

- **Left column:** The table on the left side of the join operator.
- **Right column:** The table on the right side of the join operator.
- **Operator:** the method to be used to join the tables for the result set.

Press **Next** to Continue.

This table illustrates the effect of the operators on the query result set:

Table 154: Effect of the join operators on the query result set

Operator	Description	Example WHERE clause generated for BDL code
INNER JOIN	Result set will contain columns from only the rows in the two tables where the specified column value matches.	WHERE orders.userid = account.userid
LEFT OUTER JOIN	Result set will contain columns from all the rows in the left table, joined with columns from the rows in the right table where the specified column value matches. Where there is no matching value, the columns from the right	WHERE (orders.userid = account.userid OR account.userid IS NULL)

Operator	Description	Example WHERE clause generated for BDL code
	table in the result set will contain nulls.	
RIGHT OUTER JOIN	Result set will contain columns from all the rows in the right table, joined with columns from the rows in the left table in which the specified column value matches. Where there is no matching value, the columns from the left table in the result set will contain nulls.	<pre>WHERE (orders.userid = account.userid OR orders.userid IS NULL)</pre>
FULL OUTER JOIN	Result set will contain columns from all the rows in the left table, and columns from all the rows in the right table, joined on the specified column value. Where there is no match in one of the tables, the columns from that table will contain nulls.	<pre>WHERE (orders.userid = account.userid OR account.userid IS NULL OR orders.userid IS NULL)</pre>

Result Set Page

Select the database table fields for the result set of the SQL query. The data in the result set is passed by the BDL REPORT function to the Genero Report Engine and the Report Definition file (4rp).

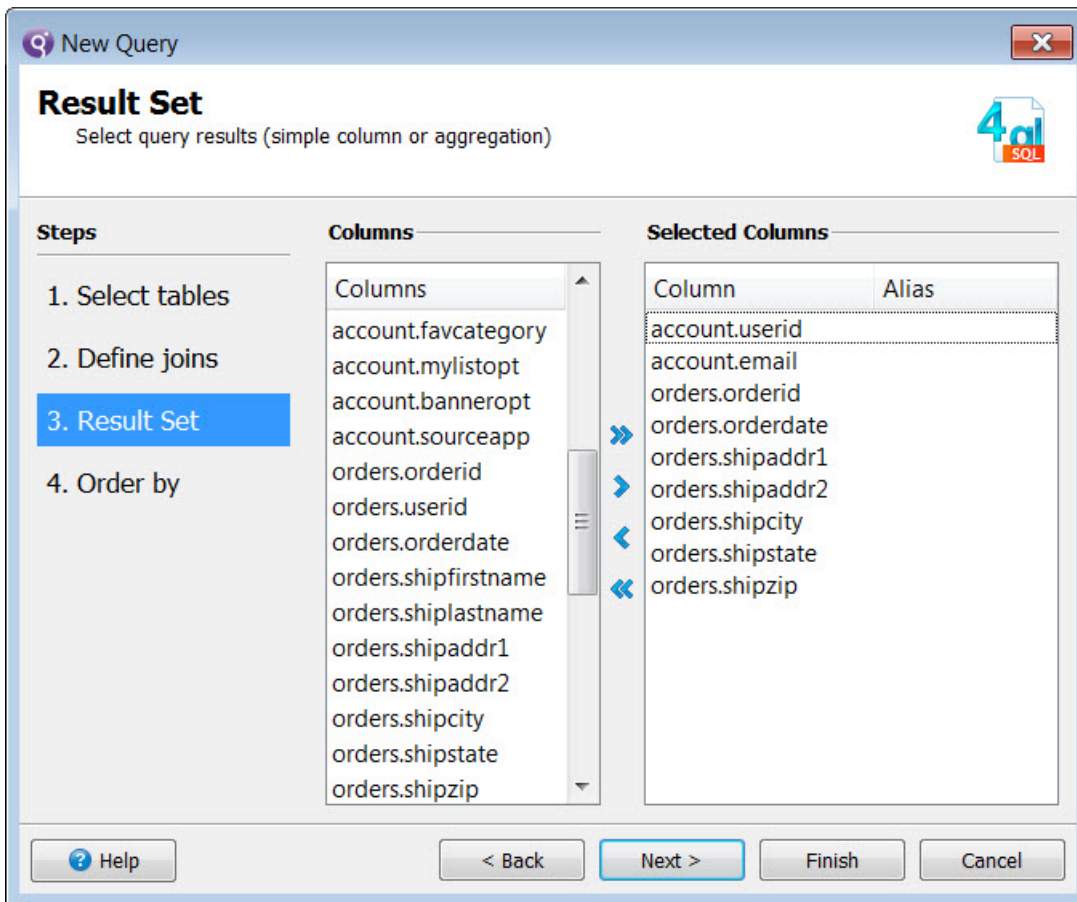


Figure 320: New Query wizard, Result Set page

Press **Next** to Continue.

Order By Page

Select the column(s) by which the result set should be sorted.

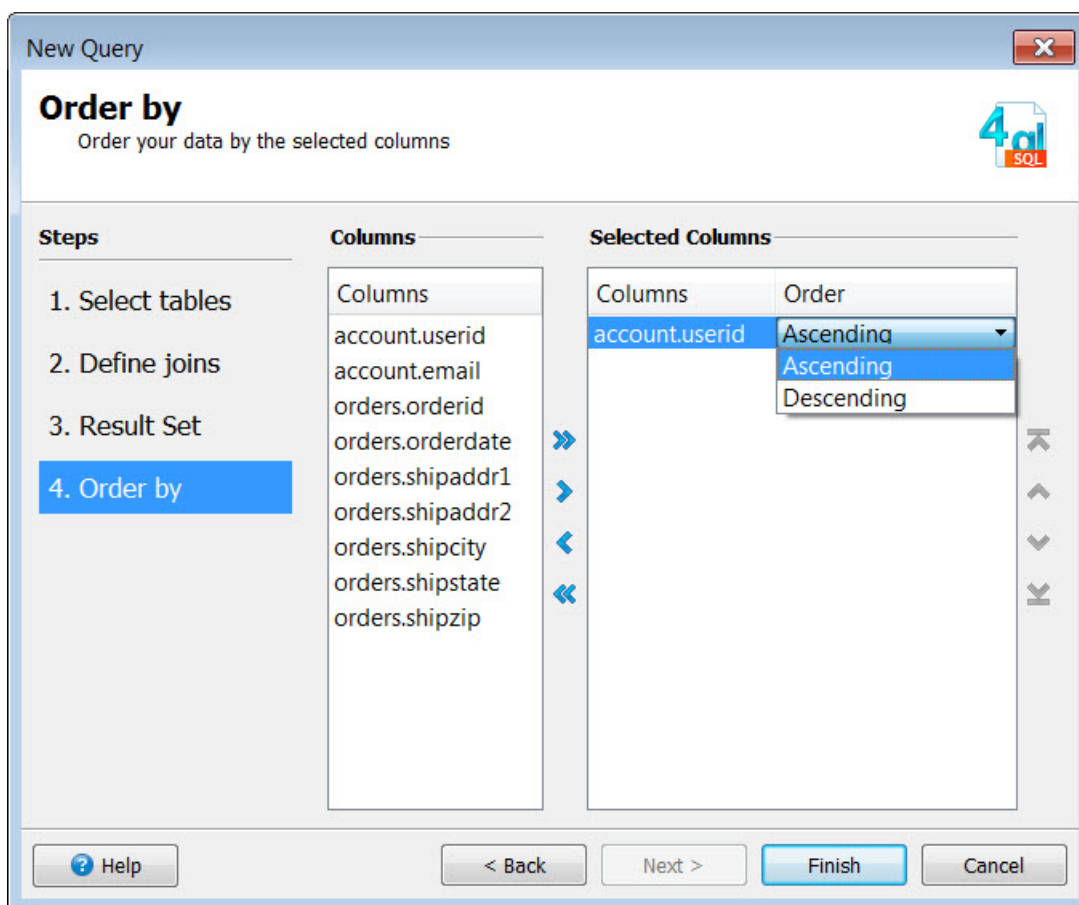


Figure 321: New Query wizard, Order By page

Press **Finish** to generate the code and close the Report Data Wizard.

The Wizard creates a BDL file (4gl) containing the generated code, which you can save as part of your application files.

Example Wizard code

This is the code that was generated by the example covered in [Using the Report Data Wizard](#) on page 581

```

SCHEMA officestore

TYPE officestore_data RECORD
  account RECORD
    userid LIKE account.userid,
    email LIKE account.email
  END RECORD,
  orders RECORD
    orderid LIKE orders.orderid,
    orderdate LIKE orders.orderdate,
    shipaddr1 LIKE orders.shipaddr1,
    shipaddr2 LIKE orders.shipaddr2,
    shipcity LIKE orders.shipcity,
    shipstate LIKE orders.shipstate,
    shipzip LIKE orders.shipzip
  END RECORD
END RECORD

FUNCTION run_officestore_to_handler(handler)

```

```

DEFINE
  data officestore_data,
  handler om.SaxDocumentHandler
DECLARE cur CURSOR FOR
  SELECT
    account.userid,
    account.email,
    orders.orderid,
    orders.orderdate,
    orders.shipaddr1,
    orders.shipaddr2,
    orders.shipcity,
    orders.shipstate,
    orders.shipzip
  FROM
    account,
    orders
  WHERE
    orders.userid = account.userid
  ORDER BY
    account.userid

START REPORT officestore_report TO XML HANDLER handler
FOREACH cur INTO data.*
  OUTPUT TO REPORT officestore_report(data.*)
END FOREACH
FINISH REPORT officestore_report
CLOSE cur
END FUNCTION

REPORT officestore_report(data)
  DEFINE
    data officestore_data
  ORDER EXTERNAL BY
    data.account.userid

  FORMAT
    ON EVERY ROW
    PRINT data.*
END REPORT

```

Note:

- Lines **3-17** define a user-type named **office_data**, which is a RECORD consisting of sub-records for the fields that were selected in the [Result Set](#) page from the tables selected in the [Table Selection](#) page.
- Lines **19-48** define the **run_officestore_to_handler** function.
 - Lines 23-40 declare a cursor for the SQL SELECT statement to retrieve the data. The table join and the sort order were specified in the [Define Joins](#) and [Order By](#) pages.
 - Lines 42-46 fetch the report data and send it to the REPORT function.
- Lines **50-59** define the REPORT function, which outputs the data to the Genero Report Engine.

See [The BDL File](#) for additional information about the BDL functions and statements used in conjunction with reports.

Using the example code

You can create a report application using the generated BDL file, or the file can be incorporated as part of a larger application. We recommend that you use a different BDL file to contain the additional code that is required, rather than making changes in the generated file. This allows you to re-execute the Report Data Wizard without risking the loss of your changes.

1. Create the Report Data Definition file

Create the `rdd` file that is used to populate the Data View of the Report Design Document, by executing this from the command line:

```
fglcomp --build-rdd <generated_file>4gl
```

Add the resulting `<sourcefile>.rdd` file to your application files.

2. Design the Report

Specify the `rdd` file on the Data View page. When your design is complete, save the report definition file (`4rp`) as part of your application. See [Designing a Report](#) for additional information about the design process.

3. Add Additional BDL Code to the program

a. To configure the report, call the mandatory report API functions, which identify the report definition file and create the XML handler for the report:

- [fgl_report_loadCurrentSettings](#) accepts the name of the Report Design document (`4rp`) as a parameter. The return value is a boolean indicating whether the load of the settings from the `4rp` file was successful.
- [fgl_report_commitCurrentSettings](#) returns a `SaxDocumentHandler` object.

Additional report API functions can be called to [change the default output options](#), if needed.

b. To connect to the Database, add a **CONNECT TO** or **DATABASE** statement to the program.

c. Call the **run_<dbname>_to_handler** function (which is in the file generated by the Report Data Wizard) to execute the report.

Example MAIN function

This simple function illustrates [step 3](#):

```
MAIN

  DEFINE handler om.SaxDocumentHandler -- report handler

  DATABASE officestore --connect to the database

  --call the mandatory functions that configure the report
  IF fgl_report_loadCurrentSettings("myreport.4rp") THEN -- if file loaded
OK
    LET handler = fgl_report_commitCurrentSettings()      -- commit the
                                                         -- file settings
  ELSE
    EXIT PROGRAM
  END IF

  -- run the report by calling the report driver contained in your
  -- generated function
  IF handler IS NOT NULL THEN
    CALL run_officestore_to_handler(handler)
  END IF

END MAIN
```

Modifying the output

- [Change report output options](#) on page 589
- [Using localized strings](#) on page 596


```
LET handler = fgl_report_commitCurrentSettings() -- commit
changes mandatory
END IF
```

Modify report output examples

- **Send output directly to the printer** - you can send the output to a printer without previewing by calling the [fgl_report_selectDevice](#) function as part of your report configuration function with the parameter "Printer":

```
CALL fgl_report_selectDevice("Printer")
```

Note: For other output formats, the parameter can be one of: "Printer", "PDF", "Image", "SVG", "HTML", "XLS" or "XLSX" (output to an Excel spreadsheet), "RTF" (output in Microsoft™ RTF format)

- **Set page margins** - the logical margins of the report page are read from the 4rp file (Report Design document.) The [fgl_report_setPageMargins](#) function overrides those margins. The syntax is:

```
CALL fgl_report_setPageMargins (topMargin String, bottomMargin String,
leftMargin String, rightMargin String)
```

```
CALL fgl_report_setPageMargins (".05cm", ".05cm", ".05cm", ".05cm")
```

- **Set Print Quality** - the [fgl_report_setPrinterQuality](#) function is used to control the quality used by the printer. By default this is not set. Setting this option reduces the set of usable printers to those matching it.

One of these values are passed to the function as a parameter: draft|high|normal. The value is passed as a String.

```
CALL fgl_report_setPrinterPrintQuality("high")
```

- **Print Double-sided** - the function [fgl_report_setPrinterSides](#) is used to specify whether the report pages should be one-sided or two-sided. By default this is not set. Setting this option reduces the set of usable printers to those matching it.

One of these values are passed to the function as a parameter. The value is passed as a String:

- **one-sided:** Each consecutive print-stream page is printed on the same side of consecutive media sheets.
- **two-sided-short-edge:** Consecutive pairs of print stream pages are printed on the front and back sides of consecutive media sheets. The orientation of the pages is correct as if the pages were to be bound along the short edge of the paper.
- **two-sided-long-edge:** Consecutive pairs of print stream pages are printed on the front and back sides of consecutive media sheets. The orientation of the pages is correct as if the pages were to be bound along the long edge of the paper.

```
CALL fgl_report_setPrinterSides("two-sided-long-edge")
```

- **Select a Paper Tray** - function [fgl_report_setPrinterMediaTray](#) controls what tray of the printer to use. This function and the functions [fgl_report_setPrintermediaTray](#) and [fgl_report_setPrintermediaName](#) are mutually exclusive. Setting this option reduces the set of usable printers to those matching it.

One of these values are passed to the function as a parameter: bottom|envelope|large-capacity|main|manual|middle|side|top. The value is passed as a String.

```
CALL fgl_report_setPrinterMediaTray("top")
```

Support for custom paper sources

On Windows™ systems the reporting API function `fgl_report_setSVGPaperSource()` can be used to identify a custom paper source for your report. This function must be used in conjunction with these functions:

- `fgl_report_setSVGPrinterName(printerName)` - to specify the printer
- `fgl_report_configureSVGPreview("PrintOnNamedPrinter")` - to bypass the preview and print silently on the named printer

The printerinfo Script

On Windows™ systems the script `$GREDIR/bin/printerinfo` is provided to list the supported paper sources that can be changed programmatically using this function. This script is executed from the command line.

The output of the script lists the device-specific source names in the left column, with the general constant in brackets on the right. Either can be used to identify the paper source in calls to `fgl_report_setSVGPaperSource()`:

This excerpt lists seven paper sources for the printer "Balzac":

```

PRINT SERVICES
Found 9 print services
  Service no 1 Printer : Balzac
    Service supports the following 18 paper sizes
      Letter 8.5 x 11 inches, 215.9 x 279.4 mm
      Legal 8.5 x 14 inches, 215.9 x 355.6 mm
      Executive 7.5 x 10 inches, 190.5 x 254 mm
      A4 (210 x 297 mm, 8.26 x 11.69 inches)
      A5 (148 x 210 mm)
      B5 (176 x 250 mm, 6.93 x 9.84 inches)
      Folio (210 x 330 mm)
      Comm10E (105 x 241 mm, U.S. Common 10 Envelope)
      DLE (110 x 220 mm)|
      Custom Unknown, or a user defined size (9 formats)
    Service supports the following 7 paper sources
      Automatically Select (FormSource)
      Auto Select (Auto)
      Tray 1 (OnlyOne)
      Tray 2 (Lower)
      Multi-Purpose Feeder (Cassette)
      Manual Paper (Manual)
      Manual Envelope (EnvelopeManual)

```

Figure 322: printerinfo output

Example:

```

CALL fgl_report_setSVGPrinterName("Balzac")
CALL fgl_report_setSVGPaperSource("Tray 2")
CALL fgl_report_configureSVGPreview("PrintOnNamedPrinter")

```

This excerpt of the `printerinfo` output is for a printer that has 27 paper sources. Since there are only 15 predefined constants for the paper sources, most of the paper sources are identified by integer constants, which can be used in calls to `fgl_report_setSVGPaperSource()` Using the integer constant can prevent encoding

issues when the device-specific name contains non-ASCII characters, such as "Sélection automatique".

```

Service no 3 Printer : HPCLJ4650DTN (Default Printer)
Service supports the following 19 paper sizes
Letter 8.5 x 11 inches, 215.9 x 279.4 mm
Legal 8.5 x 14 inches, 215.9 x 355.6 mm
Executive 7.5 x 10 inches, 190.5 x 254 mm
A3 (297 x 420 mm)
A4 (210 x 297 mm, 8.26 x 11.69 inches)
-----
A5 (148 x 210 mm)
B4 (250 x 353 mm)
B5 (176 x 250 mm, 6.93 x 9.84 inches)
Comm10E (105 x 241 mm, U.S. Common 10 Envelope)
DLE (110 x 220 mm)
Custom Unknown, or a user defined size (9 formats)
Service supports the following 27 paper sources
Automatically Select (FormSource)
Printer autom. Selection (262)
Man. Feed in Tray 1 (261)
Tray 1 (260)
Tray 2 (259)
Tray 3 (258)
Not specified (1281)
Normal (1280)
Preprinted (1279)
Letterhead (1278)
Transparent Foil (1002)

```

Figure 323: printerinfo output

Previewing a report

The SVG report format is provided for previewing a report. A report can be previewed in a desktop application or in a browser.

Preview in a desktop app: Genero Report Viewer

To preview a report in the desktop version of Genero Report Viewer, specify "SVG" as the output device.

```
fgl_report_selectDevice("SVG")
```

Preview in a browser: Genero Report Viewer for HTML5

To preview a report in the browser, you preview using Genero Report Viewer for HTML5. Specify "browser" as the output device.

```
fgl_report_selectDevice("Browser")
```

See [Send report data to a browser](#) on page 595 for additional API functions that must be called when previewing in a browser.

Preview options

When electing to preview to the Genero Report Viewer, you can configure SVG preview options:

Preview

The report is shown in a folder tab of the main preview window.

ShowPrintDialog	This option pops up the system print dialog, allowing the user to select and configure a printer. If confirmed, the document is printed in the background.
PrintOnDefaultPrinter	This option prints the report silently on the default printer. The previewer main window is not shown.
PrintOnNamedPrinter	This option prints the report silently on a specific printer. The previewer main window is not shown.

Use the API function [fgl_report_configureSVGPreview](#). For example:

```
CALL fgl_report_configureSVGPreview("ShowPrintDialog")
```

Important: Preview options set by `fgl_report_configureSVGPreview` are only valid when previewing with Genero Report Viewer in a Desktop configuration (`fgl_report_selectDevice("SVG")`). They are not valid when previewing with Genero Report Viewer for HTML5 in a Web configuration (`fgl_report_selectDevice("Browser")`).

Additional [API functions](#) provide more SVG options:

- [fgl_report_setSVGCopies](#) - function to specify the number of copies to be printed
- [fgl_report_setSVGPageRange](#) - function to select which pages should be printed
- [fgl_report_setSVGPaperSource](#) - function to select the input source of the printer
- [fgl_report_setSVGPrinterName](#) - function to select a specific printer by name
- [fgl_report_setSVGSheetCollate](#) - function to control the collation of multiple copies

Send report data to an XML file

The demo program `OrderReport.4gl` gives the user the option to output the report data to a file (`OrderReportData.xml`) in XML format.

The Reporting API function [fgl_report_create ProcessLevelData File\(\)](#) instructs the Genero Report Engine to execute the report, outputting the data specified in the PRINT function of the REPORT program block to an XML file rather than to a Report Design Document. The output file is stored in your current project. The name of the data file is passed to the function as a String.

Changing the [SimpleReport.4gl](#) example program to use this technique, the MAIN program block becomes:

```
MAIN

  DEFINE handler      om.SaxDocumentHandler,  -- return value
         dataFile    STRING      -- name of XML file to be created

  --configure the report engine to output the XML file
  LET dataFile = "./OrderReportData.xml"
  LET handler = fgl_report_createProcessLevelDataFile(dataFile)

  -- run the report, creating the XML file
  CALL runReportFromDatabase(handler)

END MAIN
```

Send report data to an HTML file

HTML output can be selected by passing the string "HTML" to the function [fgl_report_selectDevice](#). You can use this function to output any existing report (including ASCII) as an HTML document.

Graphical elements such as bar codes are implemented by rendering them as images. These images are included in the document. An option to generate the images on disk and to use external URLs instead is provided by the function [fgl_report_configureHTMLDevice](#).

When elements overlay each other (or text strings intersect), the default behavior is to create one image from the two elements and to include the image. An exception to this occurs when the overlaid element is an image: the "background" image will be removed from the resulting document.

There is an option in the function [fgl_report_configureHTMLDevice](#) to ignore the horizontal or vertical alignment in the document. The default is FALSE for HTML document output.

Send report data to an Excel spreadsheet

The report output can be sent to an Excel spreadsheet by passing the string "XLS" or "XLSX" to the function [fgl_report_selectDevice](#). The output will be in the specified Excel format, with the XLS format limited to 65,536 rows. In contrast, for the XLSX format the document is generated with constant memory consumption. so that very large documents can be produced without exhausting heap space.

Note: The XLSX format can only be opened with newer versions of Microsoft™ Excel (2007 or later). A backward compatibility pack can be downloaded from Microsoft™; however, the XLSX format will then be subject to the 65,536 limit of the earlier versions.

Any existing report, including ASCII, can be output to the spreadsheet.

The layout of the cells (size, font color, etc.) can be predictably controlled from the Report Designer. The goal is to put the report layout into the cells of the spreadsheet efficiently. There is an option in the functions [fgl_report_configureXLSDevice](#) and [fgl_report_configureXLSXDevice](#) to ignore the vertical or horizontal alignment. The default is TRUE for Excel document output, since the emphasis is on the ability to compute the values in the cells, rather than primarily on the appearance.

When items in the report design overlap, by default the placement is preserved in the spreadsheet, but not the alignment. To make the necessary decisions, Genero Studio marks the column and row boundaries internally with tabs. If there are two consecutive tabs that have no element that bounds on them, the column or row will collapse. The implication of this is:

- Rivers of white space (empty columns, empty rows) are eliminated when possible.
- Elements that overlap may be placed in the same column, but they will still maintain their relative placement (above/below).
- Contiguous items will never be placed in the same column, they will maintain their relative placement and alignment.

The values in the cells are generated for the report, not the Excel formulas. Graphical elements such as bar codes are implemented by rendering them as images. Business charts are currently drawn as tables; we cannot create charts or pivot tables.

This code fragment illustrates the functions enabling Excel output:

```
...
CALL fgl_report_configureXLSXDevice (
  NULL, #fromPage INTEGER,
  NULL, #toPage INTEGER,
  NULL, #removeWhitespace INTEGER,
  NULL, #ignoreRowAlignment INTEGER,
  NULL, #ignoreColumnAlignment INTEGER,
  NULL, #removeBackgroundImages INTEGER,
  TRUE ) #mergePages INTEGER
CALL fgl_report_selectDevice( "XLSX" )
...
```

Creating a report with data but no titles

To obtain a spreadsheet containing only the data in a flat table:

- Create a new empty report (from the main menu, **File>>New, Reports, Empty report**).
- Configure a large custom page size, **300cm x 200cm**, for example (from the main menu, **File>>Report Properties, Paper Setting**).

- From the **Toolbox**, drag a [Stripe container](#) into the report design.
- In the **Data View**, select the button [create a table column value object](#); then, double-click the fields to select them for the report.
- In the **Structure View**, associate a [Trigger node](#) with the stripe, to specify when the contents of the stripe should print (ON EVERY ROW, for example)

Creating a report with tables that contain headings

To obtain a spreadsheet containing tables with column headings, use the standard list template (from the main menu, **File>>New, Reports, List report**). Use a large custom page size, **300 cm x 200cm**, for example. The output will contain a title for the document and headings for each column.

Creating a single-sheet document

Currently each page produces a separate Sheet in Excel. If the page breaks are not desirable, you can change the page size to a larger custom value. However, creating a huge page can adversely affect memory reclamation and performance. To avoid this problem, we recommend that you use a standard page size and set the **mergepages** parameter of the function [fgl_report_configureXLSDevice](#) or [fgl_report_configureXLSXDevice](#). This will merge the pages into a single-sheet result.

Note: If you have specified "XLS" in the call to [fgl_report_selectDevice](#) to specify the Excel format, and the resulting sheet has more than 65536 rows, the exceeding rows will spill over into extra sheets; if you have specified "XLSX" as the Excel format, the size of the sheet is not limited by available memory.

Output report data in Microsoft™ RTF format

You can use the API function [fgl_report_selectDevice\("RTF"\)](#) to output the data in the RTF Format; configure the output using the function [fgl_report_configureRTFDevice](#).

In order to prevent exhaustion of main memory when processing large documents, the processor can be instructed to swap parts of the document to a temporary disk file when the document size exceeds this threshold. Use the function [fgl_report_setRTFMemoryThreshold](#) to specify the threshold.

Send report data to a browser

You can view a report in a browser with Genero Report Viewer for HTML5. Specific reporting API functions must be included in your program to display to this client.

When you start the demo program `OrderReport.4gl`, the application gives the user the option to output the report data to the Genero Report Viewer. If the Genero Studio configuration is set to use the Web configuration, this option displays the report to a browser, using Genero Report Viewer for HTML5.

To do this, the device must be set to "Browser".

```
CALL fgl_report_selectDevice("Browser");
```

When the device is set to Browser, some additional APIs must be called. You must set the browser directory and URL, and you must set the font directory and URL. For example:

```
IF outputformat=="Browser" THEN
  LET uuid=security.RandomGenerator.CreateUUIDString()
  CALL
  fgl_report_setBrowserDocumentDirectory(fgl_getenv("GRE_PRIVATE_DIR")||"/"||
  uuid)
  CALL fgl_report_setBrowserFontDirectory(fgl_getenv("GRE_PRIVATE_DIR"))
  CALL
  fgl_report_setBrowserDocumentDirectoryURL(fgl_getenv("GRE_PRIVATE_URL_PREFIX")||"/"||
  uuid)
  CALL
  fgl_report_setBrowserFontDirectoryURL(fgl_getenv("GRE_PRIVATE_URL_PREFIX"))
END IF
```

The report files

When you select Browser as the output for your report, the report files are written to the document directory.

If you are using a private directory (*GRE_PRIVATE_DIR* and *GRE_PRIVATE_URL_PREFIX*), only the current session will be able to view the report, and the report will be deleted when the session ends.

If you are using a public directory (*GRE_PUBLIC_DIR* and *GRE_PUBLIC_URL_PREFIX*), the report can be shared and bookmarked.

Warning: The environment variables *GRE_PRIVATE_DIR*, *GRE_PRIVATE_URL_PREFIX*, *GRE_PUBLIC_DIR*, and *GRE_PUBLIC_URL_PREFIX* are not documented. You should never explicitly set these variables.

Create multi-page ISO reports

For reports printed on ISO and JIS-sized pages, you can configure the output to print several logical pages per physical page. Use these [Reporting API functions](#) to change the output options:

- [fgl_report_selectLogicalPageMapping](#) - use the parameter **multipage** to change the logical page mapping, allowing the printing of multiple pages per physical page.
- [fgl_report_configureMultipageOutput](#) - to configure the required number of pages. The parameters are:
 1. **pageExponent** String - Specifies the number of pages to print. The actual number of pages is calculated by multiplying the page exponent number that you specify by 2. In the example there will be $2 * 2 = 4$ pages per physical page.
 2. **isoNumber** Integer - Specifies the ISO number. This parameter is optional (indicated by passing a null value). If no value is specified, the page size of the logical page is taken from the *4rp* file (if specified). In the example, 4 refers to ISOA4.
 3. **portrait** boolean - TRUE specifies that the page is in portrait orientation; FALSE= landscape. This parameter is optional (indicated by passing a null value). If no value is specified, the orientation of the logical page is taken from the *4rp* file.

Example function:

```
FUNCTION configure_report()
  DEFINE handler om.SaxDocumentHandler          -- configure report
engine
  IF fgl_report_loadCurrentSettings(NULL) THEN  -- there is no 4rp file
    -- change default output options
    CALL fgl_report_selectLogicalPageMapping("multipage")
    CALL fgl_report_configureMultipageOutput(2,4,TRUE)
    LET handler = fgl_report_commitCurrentSettings() -- commit settings
  ELSE
    EXIT PROGRAM
  END IF
END FUNCTION
```

Using localized strings

- [Localized Strings](#)
 - [The Source String File](#)
 - [Using the translate function](#)
 - [Entries in fglprofile](#)
- [Finding the Translated String](#)

Localized strings

The **Localized Strings** feature allows:

- the **captions** (titles of report objects) to be customized.

- The translation of the [text](#) property for Word Boxes and WordWrap Boxes to be customized.

This feature can be used to implement internationalization in your application, or to use site-specific text (for example, when business terms change based on territory).

You could have multiple external string files, each containing the translated string in a different language. The DVM (runtime system) searches the files in order to assign text in the report definition. The localized text replaces the captions and other specified text at runtime, choosing the correct string file based on entries in the file [fglprofile](#).

The Source String file

A text file, with a `str` extension, defines the string to be displayed for the report object captions or the specified Word Box or WordWrap Box text. You can check the Text property of the report object to determine the exact string to be replaced.

Syntax:

```
"identifier" = "string"
```

where:

- *identifier* is the value of the Text property of the caption or label that you want to localize
- *string* is the text that you want to be displayed instead

The backslash "\ " is accepted as the escape character, to define non-printable characters.

Example file `captions_en.str`:

```
"Userid" = "Client ID"
"Shipcity" = "City"
```

At runtime, the text "Userid" or "Shipcity" would be displayed in the report as "Client ID" and "City".

This source string file should be added to the Genero Studio Project Manager application node for your report program, so it can be compiled when the application is built. The compiled file (42s) is not linked into the Genero executable.

Note: Although you can call the API function [fgl_report_setCallbackLocalization\(\)](#) in your 4GL source code to specify that the caption of the report object be used instead, we recommend using localized strings.

Using the translate function

Entries in the source string file can also be used by the **translate** PXML function to replace text in an expression that sets a property value. For example, if you set the text of a Word Box) to this value:

```
"hello "+"world".translate()
```

and you have this entry in the `str` file:

```
"world" = "universe"
```

then the final text displayed on your WordBox will be "hello universe".

Entries in fglprofile

You can specify the compiled String File, or a list of these files, with entries in an `fglprofile` configuration file. For example:

```
fglrun.localization.warnKeyNotFound= false      -- don't display warnings on
errors
fglrun.localization.file.count = 1             -- number of string files
fglrun.localization.file.1.name = "captions_en.42s" -- name of the string
file
```

See the FGLPROFILE topics in the *Genero Business Development Language User Guide* for complete information about the `fglprofile` file.

See the Localized Strings topics in the *Genero Business Development Language User Guide* for complete information and examples.

Finding the translated string

The search behavior that Report Writer uses to find a string in a compiled string file (42s) is similar to that of Genero BDL. Once the compiled string files are located, Report Writer uses the first occurrence of the string that it finds. The 42s files are sought in this order:

1. Any localization (42s) files specified in a BDL configuration file. First, the current directory, and then all directories in `DBPATH` or `FGLRESOURCEPATH`, are searched for the specified 42s files. The precedence of BDL configuration files is:
 - a. the BDL configuration file, `<FGLDIR>/etc/fglprofile`.
 - b. the BDL configuration file specified by the `FGLPROFILE` environment variable, if set.
 - c. any BDL configuration file having the same name as the report program, and stored in the directory `<FGLDIR>/defaults`.
2. Any localization (42s) file that has the same name as the report definition file, minus the 4rp extension. First, the current directory, and then all directories listed in the `DBPATH` or `FGLRESOURCEPATH` environment variable, are searched for the `<report-name>.42s` file.
3. Any localization (42s) file that has the same name as the report application, minus the 42r extension. First, the current directory, and then all directories in `DBPATH` or `FGLRESOURCEPATH` are searched for the `<application-name>.42s` file.

Change localization settings at runtime

Use the `fgl_report_configureLocalization` function to change the directory where the program seeks the translation files and to change the formatting pattern for numerical data at runtime.

Change the language directory

Change the language directory in which the translation files (42s) are sought with the `fgl_report_configureLocalization` function. The names of the translation files (`captions.42s` and `mappings.42s`) remain constant, regardless of what language is selected. It is the directory that is changed by the API call.

```
$echo $FGLPROFILE
fglprofile
$cat $FGLPROFILE
fglrun.localization.file.count = 2
fglrun.localization.file.1.name = "captions.42s"
fglrun.localization.file.2.name = "mappings.42s"

$find translation_files
translation_files
translation_files/en
translation_files/en/captions.str
translation_files/en/captions.42s
```

```

translation_files/fr
translation_files/fr/captions.str
translation_files/fr/captions.42s
translation_files/common
translation_files/common/mappings.str
translation_files/common/mappings.42s

$cat test.4gl
MAIN
...
#select French translation
  LET handler=configureReport("translation_files/fr:translation_files/
common");
  START REPORT invoice TO XML HANDLER handler
...
END MAIN
...
FUNCTION configureReport(resourcePath)
...
  LET handler=fgl_report_loadCurrentSettings("localizable_invoice.4rp")
...
  CALL fgl_report_configureLocalization(NULL,resourcePath,NULL,NULL)
...
  RETURN fgl_report_commitCurrentSettings()
END FUNCTION
...

```

Change the formatting pattern for numerical data

Change the formatting pattern for numerical data at runtime with the `fgl_report_configureLocalization` function.

```

$echo $DBFORMAT
$:,:.:
$cat test.4gl
MAIN
...
#select German number formatting
  LET handler=configureReport(".:,:EUR");
  START REPORT invoice TO XML HANDLER handler
...
END MAIN
...
FUNCTION configureReport(numberFormat)
...
  LET handler=fgl_report_loadCurrentSettings("invoice.4rp")
...
  CALL fgl_report_configureLocalization(NULL,NULL,numberFormat,NULL)
...
  RETURN fgl_report_commitCurrentSettings()
END FUNCTION
...

```

GRW reference for Genero BDL applications

- [Reporting API Functions](#) on page 599
- [Pivot table library](#) on page 653

Reporting API Functions

Genero Report Writer provides a variety of reporting API functions for use with Genero BDL applications.

- [General](#)

- [Mandatory report functions](#)
 - [fgl_report_loadCurrentSettings\(\)](#)
 - [fgl_report_commitCurrentSettings\(\)](#)
 - [fgl_report_loadAndCommit\(\)](#)
 - [Usage](#)
- [Additional Functions to Change Default Report and Output options](#)
 - [Data](#)
 - [Form](#)
 - [Page](#)
 - [Output](#)
 - [Printer](#)
- [Additional functions to set Report Metadata for compatibility reports](#)
- [Additional functions to introspect reports at runtime \(librdd\)](#)

Overview

The Genero Report Writer reporting API for Genero BDL relies upon several files.

1. The `libgre.42m` file contains the Genero BDL functions that handle Genero Report Writer output and other features. There are [mandatory functions](#) that must be called in your Genero BDL report application, and there are [optional functions](#) that allow you to change the output format, output features and printer features.
2. These files contain some functions that are used internally by Genero Report Writer:
 - `CaptionCustom.42m`
 - `CompatCustom.42m`
 - `EncodingCustom.42m`
 - `isotools.42m`

The compiled versions of these files are provided in `<GREDIR>\lib`.

Important: These compiled files are linked in the `libgre.42x` library, which must be listed in the **external dependencies property** of any Genero Studio Project Manager application node that uses Genero Report Writer. This file should be listed by name only, without a path. Click the ... button in the Value column of the Properties View to open the Edit List window for that property.

Error Handling

In case of an error, functions from this library will write error messages to stdout using the `DISPLAY` statement. The error condition is indicated by a return value.

Verbosity level

Depending on the value of the environment variable "DEBUGLEVEL" Some of the functions issue warnings and other useful debugging information. Currently any value greater than zero will cause debug output. Debug information is also output using the `DISPLAY` statement.

Mandatory functions

These functions are required in the Report Driver section of the Genero BDL file associated with a Genero BDL reporting application.

See [Usage: load and commit](#) on page 602.

Table 155: Mandatory functions

Function	Description
<code>fgl_report_loadCurrentSettings (</code>	Loads the report definition and configures the in-memory settings accordingly.

Function	Description
<code>reportFileName</code> STRING) RETURNING <code>ok</code> INTEGER	Note: The <code>fgl_report_loadAndCommit</code> function may be used in place of the other mandatory functions if no changes to the default settings are required.
<code>fgl_report_commitCurrentSettings()</code> RETURNING <code>driver</code> <code>om.SaxDocumentHandler</code>	Configures the Genero Report Engine based on the initial report definition and any subsequent function calls. Note: The <code>fgl_report_loadAndCommit</code> function may be used in place of the other mandatory functions if no changes to the default settings are required.
<code>fgl_report_loadAndCommit</code> (<code>reportFileName</code> STRING) RETURNING <code>handler</code> <code>om.SaxDocumentHandler</code>	Function that loads a 4rp file and configures the report engine to execute the report.

`fgl_report_loadCurrentSettings`

Loads the report definition and configures the in-memory settings accordingly.

Syntax

```
fgl_report_loadCurrentSettings (
    reportFileName STRING)
RETURNING ok INTEGER
```

1. `reportFileName` - Name of the 4rp report file to process (extract the settings information). The settings are made using the [Configuration menu](#) of the Report Design page. If a relative path is specified, then it is first converted to an absolute path against the current working directory, then against FGLRESOURCEPATH, and finally against DBPATH.

The value denotes a path on the machine where the Genero Report Engine is running. In the case of distributed processing, this may be a different machine than the machine running the dynamic virtual machine (DVM). When running a legacy Genero BDL report using Report Writer, the `reportFileName` should be NULL.

This indicates the ASCII output from the BDL file is to be written to the "SVG" device in preview mode.

2. `ok` - Returns TRUE if no error occurred.

Important: If your files are loaded asynchronously, a successful call to the `fgl_report_loadCurrentSettings()` function does not guarantee that the file has loaded successfully. If your system is configured to load files asynchronously, it is recommended that you call `fgl_report_getErrorStatus()` after every call to START REPORT, OUTPUT TO REPORT, and FINISH REPORT.

Usage

This function loads the specified 4rp file and configures the current in-memory settings accordingly. Calling this function is a required prerequisite to calling `fgl_report_commitCurrentSettings()`.

The call sequence is the following:

```
CALL fgl_report_loadCurrentSettings(filename)
--- Optional functions to change default settings
```

```
CALL fgl_report_commitCurrentSettings()
```

See [Usage](#).

`fgl_report_commitCurrentSettings`

Configures the Genero Report Engine based on the initial report definition and any subsequent function calls.

Syntax

```
fgl_report_commitCurrentSettings()
RETURNING driver om.SaxDocumentHandler
```

1. *driver* - the created XMLhandler.

Usage

This function configures the report engine based on the current settings in the `4rp` file that have previously been loaded by a call to [fgl_report_loadCurrentSettings\(\)](#), and have possibly been modified.

It is mandatory to call `fgl_report_commitCurrentSettings` to set up the report engine, unless there are no changes to the default settings; in that case, [fgl_report_loadAndCommit](#) can be substituted for both of the functions.

See [Usage](#).

`fgl_report_loadAndCommit`

Function that loads a `4rp` file and configures the report engine to execute the report.

Syntax

```
fgl_report_loadAndCommit (
  reportFileName STRING)
RETURNING handler om.SaxDocumentHandler
```

1. *reportFileName* - Name of the `4rp` report file to process (extract the settings information). The settings are made using the [Configuration menu](#) of the Report Design page. If a relative path is specified, then it is first converted to an absolute path against the current working directory, then against FGLRESOURCEPATH, and finally against DBPATH.

The value denotes a path on the machine where the Genero Report Engine is running. In the case of distributed processing, this may be a different machine than the machine running the dynamic virtual machine (DVM). When running a legacy Genero BDL report using Report Writer, the *reportFileName* should be NULL.

This indicates the ASCII output from the BDL file is to be written to the "SVG" device in preview mode.

2. *handler* - Null if an error occurred.

Usage

Use this function when no change of the default settings of the specified report is required.

This function is a convenience function, replacing the need to call both a load function and a commit function.

```
CALL fgl_report_loadCurrentSettings(reportFileName)
CALL fgl_report_commitCurrentSettings()
```

See [Usage](#).

Usage: load and commit

The functions `fgl_report_loadCurrentSettings` and `fgl_report_commitCurrentSettings` work together to configure the report.

The call sequence is:

```
IF fgl_report_loadCurrentSettings(reportfilename) THEN -- function returns
  TRUE
  -- optional functions to change settings would go here
  LET repandler = fgl_report_commitCurrentSettings() -- function returns
  the
  -- SAXDocumentHandler
END IF
```

These functions must appear in the Report Driver section of the BDL file associated with the report. If no settings are to be changed, the function `fgl_report_loadAndCommit` may be substituted for these two functions.

See [Writing the Genero BDL report program](#) on page 564 for complete examples.

Functions to change reporting options

Use these functions to change the default settings for a report, and to configure output and printers. The functions are provided as part of the `libgre.42x` library.

Table 156: Environment Configuration functions

Function	Description
<code>fgl_report_setEnvironment(values om.SaxAttributes)</code>	Specifies variable values in the private environment of the report.

Table 157: Data functions

Function	Description
<code>fgl_report_createProcessLevelDataFile(dataFileName STRING) RETURNING driver om.SaxDocumentHandler</code>	Configures the report execution to output an XML datafile.
<code>fgl_report_runFromXML(dataFileName STRING) RETURNING ok INTEGER</code>	Replays the report from an XML file.
<code>fgl_report_runReportFromProcessLevelDataFile(out om.SaxDocumentHandler, fileName STRING) RETURNING ok INTEGER</code>	Runs a report from a process level file.
<code>fgl_report_setProcessLevelDataFile(dataFileName STRING)</code>	Configures the report execution to output an XML datafile in addition to the regular processing.

Table 158: Form functions

Function	Description
<code>fgl_report_findResourcePath(reportName STRING)</code>	Returns the path to a resource searching first FGLRESOURCEPATH, then FGLDBPATH and finally DBPATH. (deprecated!)

Table 159: Page Functions

Function	Description
<code>fgl_report_configureLabelOutput(paperWidth STRING, paperHeight STRING, labelWidth STRING, labelHeight STRING, labelsPerRow INTEGER, labelsPerColumn INTEGER)</code>	Configures the physical layout of a label page.
<code>fgl_report_configureMultipageOutput(pageExponent STRING, isoNumber INTEGER, portrait BOOLEAN)</code>	Configure the multipage output for ISO or JIS formats.
<code>fgl_report_configurePageSize(pageWidth STRING, pageHeight STRING)</code>	Set values for the page height and page width for Genero BDL ASCII reports being run in graphical mode.
<code>fgl_report_selectLogicalPageMapping(mapping STRING)</code>	Configures the mapping of logical pages to physical pages.
<code>fgl_report_setPageMargins(topMargin STRING, bottomMargin STRING, leftMargin STRING, rightMargin STRING)</code>	Configure the logical margins of a report.
<code>fgl_report_setPageSwappingThreshold(value INTEGER)</code>	Sets the threshold for page-to-disk swapping.
<code>fgl_report_setPaperMargins(topMargin STRING, bottomMargin STRING, leftMargin STRING, rightMargin STRING)</code>	Configure the physical margins of a report.

Table 160: Output functions

Function	Description
<pre>fgl_report_configureAutoformatOutput (fontName STRING, fontSize INTEGER, fidelity BOOLEAN, reportTitle STRING, fieldNamePatterns STRING, systemId STRING)</pre>	Configure the output when auto-formatting is enabled..
<pre>fgl_report_configureCompatibilityOutput (pageWidthInCharacters INTEGER, fontName STRING, fidelity BOOLEAN, reportName NULL, reportCategory STRING, systemId STRING)</pre>	Configure the output for BDL ASCII reports (compatibility reports) being run in graphical mode using Genero Report Writer.
<pre>fgl_report_configureHTMLDevice (fromPage INTEGER, toPage INTEGER, embedImages INTEGER, imageGenerationDirectory STRING, imageURLPrefix STRING, removeWhitespace INTEGER, ignoreRowAlignment INTEGER, ignoreColumnAlignment INTEGER, removeBackgroundImages INTEGER)</pre>	Configure the HTML output.
<pre>fgl_report_configureImageDevice (antialiasFonts BOOLEAN, antialiasShapes BOOLEAN, monochrome BOOLEAN, fromPage INTEGER, toPage INTEGER, fileType STRING, filePath STRING, fileNamePrefix STRING, resolution INTEGER)</pre>	Configure the image output.
<pre>fgl_report_configureLocalization (charSet STRING, resourcePath STRING, numberFormat STRING, dateFormat STRING)</pre>	Configures the localization for the current report.
<pre>fgl_report_configureOORTFDevice (fromPage INTEGER, toPage INTEGER, imagesResolution INTEGER,</pre>	Configure RTF output for Open Office.

Function	Description
<code>imagesFormat STRING)</code>	
<code>fgl_report_configurePDFDevice(fontDirectory STRING, antialiasFonts BOOLEAN, antialiasShapes BOOLEAN, monochrome BOOLEAN, fromPage INTEGER, toPage INTEGER)</code>	Configure the PDF output.
<code>fgl_report_configurePDFFontEmbedding(preferUnicodeEncoding BOOLEAN)</code>	Configure the font embedding in PDF output.
<code>fgl_report_configureRTFDevice(fromPage INTEGER, toPage INTEGER, imagesResolution INTEGER, imagesFormat STRING)</code>	Configure RTF output for Microsoft™.
<code>fgl_report_configureSVGDevice(antialiasFonts BOOLEAN, antialiasShapes BOOLEAN, embedFonts BOOLEAN, charsetToEmbed STRING)</code>	Configure the SVG output.
<code>fgl_report_configureSVGPreview(preview STRING)</code>	Select how the document is handled by the SVG previewer.
<code>fgl_report_configureXLSDevice(fromPage INTEGER, toPage INTEGER, removeWhitespace INTEGER, ignoreRowAlignment INTEGER, ignoreColumnAlignment INTEGER, removeBackgroundImages INTEGER, mergePages INTEGER)</code>	Configure the XLS (Excel) output.
<code>fgl_report_configureXLSXDevice (fromPage INTEGER, toPage INTEGER, removeWhitespace INTEGER, ignoreRowAlignment INTEGER, ignoreColumnAlignment INTEGER, removeBackgroundImages INTEGER, mergePages INTEGER)</code>	Configure Excel output in XLSX format.
<code>fgl_report_selectDevice(</code>	Select the output device.

Function	Description
<code>device STRING)</code>	
<code>fgl_report_selectPreview(preview INTEGER)</code>	The <code>fgl_report_selectPreview</code> function determines whether the report is shown in the previewer or written to a file on disk.
<code>fgl_report_setAutoformatType(type STRING)</code>	Sets the auto-formatting type if no <code>4rp</code> template is specified.
<code>fgl_report_setBrowserDocumentDirectory(directory STRING)</code>	Configures the document directory for browser viewing.
<code>fgl_report_setBrowserDocumentDirectoryURL(directory STRING)</code>	Configures the URL by which the document directory for browser viewing will be visible from a web server.
<code>fgl_report_setBrowserFontDirectory(directory STRING)</code>	Configures the font directory for browser viewing.
<code>fgl_report_setBrowserFontDirectoryURL(directory STRING)</code>	Configures the URL by which the font directory for browser viewing will be visible from a web server.
<code>fgl_report_setCallbackLocalization(share BOOLEAN)</code>	Configure the report to use a function to retrieve localized field titles.
<code>fgl_report_setImageShrinkImagesToPageContent(value BOOLEAN)</code>	Configure image cropping.
<code>fgl_report_setImageUsePageNamesAsFileNames(value BOOLEAN)</code>	Configure image file name generation.
<code>fgl_report_setOutputFileName(fileName STRING)</code>	Configure the file location for the device output. This function works for all output formats except Image.
<code>fgl_report_setPDFImageResolution(imagesResolution INTEGER)</code>	Configure the resolution of embedded images in PDF output.
<code>fgl_report_setPDFJPEGImageEncoding(encodeImagesAsJPEG STRING, jpegQuality BOOLEAN)</code>	Configure the encoding method of embedded images in PDF output.
<code>fgl_report_setRTFMemoryThreshold (</code>	Set the RTF memory threshold.

Function	Description
<code>memoryThreshold</code> INTEGER)	
<code>fgl_report_setSharePortWithGDC(share</code> BOOLEAN)	Configures the report engine to use the same port as the Genero Desktop Client for previewing.
<code>fgl_report_setSVGCompression(compressOutput</code> BOOLEAN)	Configure SVG compression.
<code>fgl_report_setSVGCopies(copies</code> INTEGER)	Specify the number of copies to be printed.
<code>fgl_report_setSVGOrientationRequested(orientationRequested</code> String)	Function to control the paper orientation.
<code>fgl_report_setSVGPageRange(fromPage</code> INTEGER, <code>toPage</code> INTEGER)	Select which pages should be printed.
<code>fgl_report_setSVGPaperSource(paperSource</code> STRING)	Select the input source of the printer.
<code>fgl_report_setSVGPrinterName(printerName</code> STRING)	Select a specific printer by name.
<code>fgl_report_setSVGSheetCollate (sheetCollate</code> STRING)	Control the collation of multiple copies.
<code>fgl_report_setXLSMergeCells (mergeCells</code> BOOLEAN)	Configure cell merging in XLS (Excel) output.
<code>fgl_report_setXLSXMergeCells (mergeCells</code> BOOLEAN)	Configure cell merging in XLSX (Excel) output.
<code>fgl_report_stopGraphicalCompatibilityMode()</code>	Restore text-based report output.

Table 161: Printer functions

Function	Description
<code>fgl_report_setPrinterChromaticity(chromaticity</code> STRING)	Control color selection of the printer.
<code>fgl_report_setPrinterCopies(</code>	Specify the number of copies to be printed.

Function	Description
<code>copies INTEGER)</code>	
<code>fgl_report_setPrinterDestinationUrl(destination STRING)</code>	Specify an alternate destination for the spooled printer formatted data.
<code>fgl_report_setPrinterFidelity(fidelity INTEGER)</code>	Select printer high fidelity mode.
<code>fgl_report_setPrinterJobImpressions(jobImpressions INTEGER)</code>	Specify the total size in number of impressions.
<code>fgl_report_setPrinterJobMediaSheets(jobMediaSheets INTEGER)</code>	Specify the total number of media sheets.
<code>fgl_report_setPrinterJobName(jobName STRING)</code>	Specify a name for the job.
<code>fgl_report_setPrinterJobPriority(jobPriority INTEGER)</code>	Specify a priority for the job.
<code>fgl_report_setPrinterJobSheets(jobSheets STRING)</code>	Control job sheet printing.
<code>fgl_report_setPrinterMediaName(mediaTray STRING)</code>	Select the type of media to use.
<code>fgl_report_setPrinterMediaSizeName(mediaSizeName STRING)</code>	Select the media size to be used.
<code>fgl_report_setPrinterMediaTray(mediaTray STRING)</code>	Select the tray of the printer.
<code>fgl_report_setPrinterName(printerName STRING)</code>	Select a specific printer by name.
<code>fgl_report_setPrinterNumberUp(numberUp INTEGER)</code>	Specify the number of print stream pages for a single side of an instance.
<code>fgl_report_setPrinterOrientationRequested(orientationRequested STRING)</code>	Control the paper orientation.
<code>fgl_report_setPrinterPageRanges(</code>	Specify the ranges of pages to print.

Function	Description
<code>pageRanges</code> STRING)	
<code>fgl_report_setPrinterPrintQuality(printQuality</code> STRING)	Control the quality used by the printer.
<code>fgl_report_setPrinterRequestingUserName(requestingUserName</code> STRING)	Specify end user's name.
<code>fgl_report_setPrinterResolution(resolution</code> INTEGER)	Specify an exact resolution for the printer.
<code>fgl_report_setPrinterSheetCollate(sheetCollate</code> STRING)	Controls the collation of multiple copies.
<code>fgl_report_setPrinterSides(sides</code> STRING)	Specify the mapping of pages on the physical media.
<code>fgl_report_setPrinterWriteToFile(file</code> STRING)	Specify a file where the report is written in postscript. (deprecated!)

Table 162: Distributed Mode functions

Function	Description
<code>fgl_report_configureDistributedEnvironment(FGLDIR</code> STRING, <code>FGLPROFILE</code> STRING, <code>FGLRESOURCEPATH</code> STRING, <code>DBPATH</code> STRING)	Configure the environment in the case of distributed processing.
<code>fgl_report_configureDistributedProcessing(host</code> STRING, <code>port</code> INTEGER)	Configure processing via a dedicated server.
<code>fgl_report_setDistributedRequestingUserName(requestingUserName</code> STRING)	Specify end user's name for the purpose of identifying log entries in the case of distributed processing.
<code>fgl_report_configureDistributedURLPrefix(urlPrefix</code> STRING)	Configures the URL prefix containing the host and optionally the port and prepended paths for previewing a document.

Using report output functions

When you load the current settings of a report, you load report output options specified in the report design document. To change how the report is output, the output functions must be called after the loading of the current settings ([fgl_report_loadCurrentSettings](#) on page 601), but before committing the current settings ([fgl_report_commitCurrentSettings](#) on page 602).

In this example, the Genero BDL code fragment uses the [fgl_report_selectDevice](#) on page 626 function to change the output device to PDF and the [fgl_report_selectPreview](#) on page 628 function to select to preview the report.

```

IF fgl_report_loadCurrentSettings(r_filename) THEN -- load the 4rp file
                                                    -- and continue if
    successful
    CALL fgl_report_selectDevice("PDF")           -- change to PDF
    CALL fgl_report_selectPreview(TRUE)          -- preview file using
    default previewer
    -- ADD ADDITIONAL API CALLS HERE, BEFORE YOU COMMIT CURRENT SETTINGS.
    LET handler = fgl_report_commitCurrentSettings() -- commit changes
ELSE
    EXIT PROGRAM
END IF

```

Device-specific function summary list

Device-specific configuration functions listed by device.

Table 163: Device-specific functions

Device	Available functions
HTML	fgl_report_configureHTMLDevice on page 617.
Image	fgl_report_configureImageDevice on page 615, fgl_report_setImageShrinkImagesToPageContent on page 631, fgl_report_setImageUsePageNamesAsFileNames on page 631.
OORTF	fgl_report_configureOORTFDevice on page 618, fgl_report_setRTFMemoryThreshold on page 645
PDF	fgl_report_configurePDFDevice on page 619, fgl_report_configurePDFFontEmbedding on page 620, fgl_report_setPDFJPEGImageEncoding on page 634, fgl_report_setPDFImageResolution on page 634.
Postscript	None.
Printer	fgl_report_setPrinterChromaticity on page 635, fgl_report_setPrinterCopies on page 635, fgl_report_setPrinterDestinationUrl on page 635, fgl_report_setPrinterFidelity on page 636, fgl_report_setPrinterJobImpressions on page 636, fgl_report_setPrinterJobMediaSheets on page 637, fgl_report_setPrinterJobName on page 637, fgl_report_setPrinterJobPriority on page 637, fgl_report_setPrinterJobSheets on page 638, fgl_report_setPrinterMediaName on page 638, fgl_report_setPrinterMediaSizeName on page 639, fgl_report_setPrinterMediaTray on page 639, fgl_report_setPrinterName on page 640, fgl_report_setPrinterNumberUp on page 640, fgl_report_setPrinterOrientationRequested on page 640, fgl_report_setPrinterPageRanges on page 641, fgl_report_setPrinterPrintQuality on page 641, fgl_report_setPrinterRequestingUserName on page 642, fgl_report_setPrinterResolution on page 642, fgl_report_setPrinterSheetCollate on page 643, fgl_report_setPrinterSides on page 643, fgl_report_setPrinterWriteToFile on page 644.
RTF	fgl_report_configureRTFDevice on page 620, fgl_report_setRTFMemoryThreshold on page 645.
SVG	fgl_report_configureSVGDevice on page 621, fgl_report_setSVGCompression on page 645, fgl_report_setSVGCopies on page 646,

Device	Available functions
	fgl_report_setSVGOrientationRequested on page 646, fgl_report_setSVGPageRange on page 647, fgl_report_setSVGPaperSource on page 647, fgl_report_configureSVGPreview on page 621, fgl_report_setSVGPrinterName on page 647, fgl_report_setSVGSheetCollate on page 648.
XLS	fgl_report_configureXLSDevice on page 622, fgl_report_setXLSMergeCells on page 648.
XLSX	fgl_report_configureXLSXDevice on page 623, fgl_report_setXLSXMergeCells on page 649.

`fgl_report_configureAutoformatOutput`

Configure the output when auto-formatting is enabled.

Syntax

```
fgl_report_configureAutoformatOutput (
    fontName STRING,
    fontSize INTEGER,
    fidelity BOOLEAN,
    reportTitle STRING,
    fieldNamePatterns STRING,
    systemId STRING )
```

1. *fontName* specifies the font to use.
2. *fontSize* specifies the font size to use
3. *fidelity* specifies whether or not to set the [fidelity](#) property for the produced WORDBOX objects. See [WORDBOX](#) for more information.
4. *reportTitle* - Title of the report.
5. *fieldNamePatterns* - A comma separated list of field name patterns; fields not matching any of the patterns are not printed. The patterns may contain literal characters, the ? question mark, the * star character, and character ranges, as defined for the Genero BDL MATCHES operator. The columns of the output are sorted in order of the patterns matched and within one pattern by the relative position of the field in the PRINT statement.
6. *systemId* specifies an absolute URL against which relative resources such as images in overlays are resolved.

Usage

This function is applicable when no 4rp template has been specified in the call to either [fgl_report_loadCurrentSettings](#) or [fgl_report_loadAndCommit](#), and auto-formatting with a value other than COMPATIBILITY has been selected by a call to [fgl_report_setAutoformatType](#).

All arguments to this function are optional (indicated by passing a null value).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureCompatibilityOutput`

Configure the output for BDL ASCII reports (compatibility reports) being run in graphical mode using Genero Report Writer.

Syntax

```
fgl_report_configureCompatibilityOutput (
```

```

pageWidthInCharacters INTEGER,
fontName STRING,
fidelity BOOLEAN,
reportName NULL,
reportCategory STRING,
systemId STRING)

```

1. *pageWidthInCharacters* - For reports that do not contain a RIGHT MARGIN specification this value should be set. If the report does not contain a RIGHT MARGIN specification and this value is not set, a value of 132 is assumed.
2. *fontName* - Specifies the font to use. The default is a fixed pitch font.
3. *fidelity* - Ensures that the text preview and text printout are 100% the same. The font is not embedded in the report document, it is drawn similar to an image. As a result, you may not be able to select the text in the resulting report, depending on the output format chosen (pdf, for example).
4. *reportName* - the value for this parameter is now provided internally. However, because of backwards compatibility, you must set the value to NULL when you call this function.
5. *reportCategory* - Specifies the category of the report. The value specified is passed to the overloadable callback function `compat_placePageBackground` in `$GREDIR/src/overloadables/CompatCustom.4gl`. By default, the Genero Report Engine calls `compat_placePageBackground(out,reportName,reportCategory,pageNumber)`; if the report category is "demo" or "form", specific actions are taken to the report as written in the source code of `CompatCustom.4gl`. To provide for custom report categories, create a copy of `CompatCustom.4gl` in a different path, then update `FGLLDPATH` and put the path containing your copy BEFORE the path containing the original; the DVM will load your file instead of the default. For example, you may want to place a company logo in all reports with the category "correspondence", such as an invoice, delivery receipt, and so on.
6. *systemId* - Specifies an absolute URL against which relative resources such as images in overlays are resolved.

Usage

Function to optionally configure the output for BDL ASCII reports (compatibility reports) being run in graphical mode using Genero Report Writer.

This function is applicable when no `4rp` template has been specified in the call to either [fgl_report_loadCurrentSettings](#) or [fgl_report_loadAndCommit](#). All arguments to this function are optional (indicated by passing a null value).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureDistributedEnvironment`

Configure the environment in the case of distributed processing.

Syntax

```

fgl_report_configureDistributedEnvironment (
  FGLDIR STRING,
  FGLPROFILE STRING,
  FGLRESOURCEPATH STRING,
  DBPATH STRING )

```

1. *FGLDIR* specifies the value of the environment variable `FGLDIR`. Passing NULL for this value will default to the process environment variable value.
2. *FGLPROFILE* specifies the value of the environment variable `FGLPROFILE`. Passing NULL for this value will default to the process environment variable.

3. *FGLRESOURCEPATH* specifies the value of the environment variable *FGLRESOURCEPATH*. Passing NULL for this value will default to the process environment variable value.
4. *DBPATH* specifies the value of the environment variable *DBPATH*. Passing NULL for this value will default to the environment process environment variable value.

Usage

Configure the environment. It is intended for the case of distributed processing with a server running on a different physical machine with different resource paths. It is not necessary to call this function if the daemon is running on the local machine or if the remote machine has identical resource directories.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureDistributedProcessing`
Configure processing via a dedicated server.

Syntax

```
fgl_report_configureDistributedProcessing(  
    host STRING,  
    port INTEGER )
```

1. *host* specifies the host name.
2. *port* specifies the port number.

Usage

Specify the server where the Genero Report Engine is running in server mode. The engine is started on the remote machine via the command `$GREDIR/bin/greportwriter -l port`.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureDistributedURLPrefix`
Configures the URL prefix containing the host and optionally the port and prepended paths for previewing a document.

Syntax

```
fgl_report_configureDistributedURLPrefix(  
    urlPrefix STRING )
```

1. *urlPrefix* is a URL by which the web server can be reached.

Usage

This function is applicable when previewing has been selected by a call to the function `fgl_report_selectPreview` and the Genero Report Engine (GRE) is running in distributed mode on a remote machine.

Note: This option is not needed and not taken into consideration when the GRE is running in distributed mode on the same machine as the DVM.

This function needs to be called only if the web server deviates from the default "http://HOST:8080", where "HOST" denoted the value "host" passed in the call to `fgl_report_configureDistributedProcessing(host, port)`.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureImageDevice`
Configure the image output.

Syntax

```
fgl_report_configureImageDevice (
    antialiasFonts BOOLEAN,
    antialiasShapes BOOLEAN,
    monochrome BOOLEAN,
    fromPage INTEGER,
    toPage INTEGER,
    fileType STRING,
    filePath STRING,
    fileNamePrefix STRING,
    resolution INTEGER)
```

1. *antialiasFonts* - Configures whether fonts should be rendered using antialiasing. The default value is true.
2. *antialiasShapes* - Configures whether shapes should be rendered using antialiasing. The default value is true.
3. *monochrome* - Configures whether color values should be converted to monochrome output. The default value is true.
4. *fromPage* - Selects the lower bound of the range of pages to create images for. The default value is 1.
5. *toPage* - Selects the upper bound of the range of pages to create images for. By default, images are created for all pages.
6. *fileType* - One of jpg|png|bmp|gif
7. *filePath* - Path of the target directory where the images are created
8. *fileNamePrefix* - Name prefix of the generated files (e.g. setting namePrefix to "Chart" will cause the creation of files called "Chart0001", "Chart0002", and so on.
9. *resolution* - Controls the resolution used for creating the images. If the image is later viewed unscaled on a device with the specified resolution, all items will have their correct metric length. Beware that high values may require enormous amounts of memory and the resulting files may become very large. The formula for calculating the memory consumption in bytes is $\text{resolution_in_dpi_x} * \text{page_width_in_inches} * \text{resolution_in_dpi_y} * \text{page_height_in_inches} * 3\text{byte}$ for a color image. For a page of format "letter" at 96 DPI we therefore get $96\text{DPI} * 8.5" * 96\text{DPI} * 11" * 3\text{byte} = 2.6 \text{ MB}$. At 300 DPI this is 25 MB (color) and 8 MB (grayscale). The renderer currently requires the entire page to be in memory.

Usage

This function is applicable when image output has been selected by a call to the function [fgl_report_selectDevice](#).

All arguments to this function are optional (indicated by passing a null value).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureLabelOutput`
Configures the physical layout of a label page.

Syntax

```
fgl_report_configureLabelOutput (
```

```
paperWidth STRING,
paperHeight STRING,
labelWidth STRING,
labelHeight STRING,
labelsPerRow INTEGER,
labelsPerColumn INTEGER)
```

1. *paperWidth* - width of the page (e.g. a4width)
2. *paperHeight* - height of the page (e.g. a4length)
3. *labelWidth* - Physical width of a label (e.g. 99.1mm) The value specified here should be the same or larger than the width in the 4rp file. The value is optional (indicated by passing null). In this case the width specified in the 4rp file is used.
4. *labelHeight* - Physical height of a label (e.g. 42.3mm) The value specified here should be the same or larger than the height in the 4rp file. The value is optional (indicated by passing null). In this case the height specified in the 4rp file is used.
5. *labelsPerRow* - the number of labels across
6. *labelsPerColumn* - the number of labels down

For additional information about the strings that can be used to specify the parameter values, see [Dimension Resolver](#) on page 801.

Usage

Function that configures the physical layout of a label page.

This function is applicable when selecting "label" as the mapping option by calling the function [fgl_report_selectLogicalPageMapping](#). This figures the physical layout by specifying the paper dimensions, the physical label size and the $n \times m$ layout. The physical margins (distance between page border and label) are specified by calling [fgl_report_setPaperMargins](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureLocalization`

Configures the localization for the current report.

Syntax

```
fgl_report_configureLocalization(
  charSet STRING,
  resourcePath STRING,
  numberFormat STRING,
  dateFormat STRING)
```

1. *charSet* - Specifies the encoding of the translation files (`str` and `42s` files).
2. *resourcePath* - A colon- or semicolon-delimited (Windows™) list of directories specifying the search path for compiled translation files (`42s`).
3. *numberFormat* - A formatting string to be used for number formatting. The format is compatible with the format of the Genero Business Development Language `DBFORMAT` environment variable.
4. *dateFormat* - A formatting string to be used for date formatting. The format is compatible with the format of the Genero Business Development Language `DBDATE` environment variable.

Usage

Function that configures the localization for the current report.

The function is not applicable for callback localization (See [fgl_report_setCallbackLocalization](#)).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureHTMLDevice`
Configure the HTML output.

Syntax

```
fgl_report_configureHTMLDevice(
    fromPage INTEGER,
    toPage INTEGER,
    embedImages INTEGER,
    imageGenerationDirectory STRING,
    imageURLPrefix STRING,
    removeWhitespace INTEGER,
    ignoreRowAlignment INTEGER,
    ignoreColumnAlignment INTEGER,
    removeBackgroundImages INTEGER )
```

1. *fromPage* - Selects the lower bound of the range of pages to include in the HTML document. The default value is 1.
2. *toPage* - Selects the upper bound of the range of pages to include in the HTML document. By default all pages are included.
3. *embedImages* - Specifies whether to embed images in the resulting HTML output. By default this is not the case.
4. *imageGenerationDirectory* - If images are not embedded, this property specifies the directory into which generated images are written. The directory needs to exist, it is not created. Note that the urls that are created will not take this value into account. By default, the urls that are created contain the image name only, so that the images are expected to reside in the same directory as the document. If needed, the url prefix for the generated urls can be changed with the property *imageURLPrefix*.
5. *imageURLPrefix* - If images are not embedded, this property specifies the prefix of the urls of the generated images. As an example consider that an image of the name "12345.png" is created, and that this property is set to the value "./images/"; then the "src" attribute of the generated "img" element would be set to "./images/12345.png".
6. *removeWhitespace* - Controls whether cells should be created for empty strings. By default whitespace is stripped from the document.
7. *ignoreRowAlignment* - When set, only those objects that are entirely above or entirely below each other will go in separate rows. When set, the option reduces the amount of rows, thereby losing the horizontal alignment. The placement is not changed so that stacked items remain stacked. By default row alignment is not ignored.
8. *ignoreColumnAlignment* - When set, only those objects that are entirely to the left or entirely to the right of each other will go in separate columns. When set, the option reduces the amount of columns, thereby losing the vertical alignment. The placement is not changed so that adjacent items remain adjacent. By default column alignment is not ignored.
9. *removeBackgroundImages* - Controls the behavior when an IMAGEBOX is partially obscured by another element. When set, the image is removed from the resulting document; otherwise, the handling is as in any other case of overlapping items. By default, background images are removed.

Usage

Function to configure the HTML output.

This function is applicable when HTML output has been selected by a call to the function [fgl_report_selectDevice](#). All arguments to this function are optional (indicated by passing a null value). If the HTML document should be written to a file, the general functions [fgl_report_setOutputFileName](#) and [fgl_report_selectPreview](#) are available for this purpose.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureMultipageOutput`

Configure the multipage output for ISO or JIS formats.

Syntax

```
fgl_report_configureMultipageOutput (
    pageExponent STRING,
    isoNumber INTEGER,
    portrait BOOLEAN )
```

1. *pageExponent* - Specifies the number of pages to print. The actual number of pages is calculated by the formula $\text{count}=2^{\text{pageExponent}}$.
2. *isoNumber* - Specifies the ISO number (e.g. 4 to ISOA4) This parameter is optional (indicated by passing an null value). If no value is specified, the page size of the logical page is taken from the 4rp file (if specified).
3. *portrait* - TRUE specifies that the page is in portrait orientation; FALSE= landscape. This parameter is optional (indicated by passing a null value). If no value is specified, the value of the logical page is taken from the 4rp file (if specified), or from the value specified in a call to [fgl_report_configurePageSize](#).

Usage

Function to configure the multipage output for ISO or JIS formats.

This function is applicable for ISO or JIS formats and enables the printing of several logical pages per physical page. The number of pages per page is always a power of 2.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureOORTFDevice`

Configure RTF output for Open Office.

Syntax

```
fgl_report_configureOORTFDevice (
    fromPage INTEGER,
    toPage INTEGER,
    imagesResolution INTEGER,
    imagesFormat STRING )
```

1. *fromPage* - Selects the lower bound of the range of pages to include in the RTF document. The default value is 1.
2. *toPage* - Selects the upper bound of the range of pages to include in the RTF document. Per default all pages are included.
3. *imagesResolution* - Specifies the resolution of embedded images. In addition to ImageBoxes, content from BarCodeBoxes, business charts and HTMLBoxes are embedded as images.
4. *imagesFormat* - One of : "png", "jpg". Controls the format of images embedded in the RTF document. Select jpg for compactness, png for lossless compression.

Usage

Function to configure RTF output for Open Office.

This function is applicable when RTF output has been selected by a call to the function [fgl_report_selectDevice](#). All arguments to this function are optional (indicated by passing a null value).

If the RTF document should be written to a file, the general functions [fgl_report_setOutputFileName](#) and [fgl_report_selectPreview](#) are available for this purpose.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configurePageSize`

Set values for the page height and page width for Genero BDL ASCII reports being run in graphical mode.

Syntax

```
fgl_report_configurePageSize(
    pageWidth STRING,
    pageHeight STRING)
```

1. *pageWidth* - width of the page (e.g. a4width).
2. *pageHeight* - height of the page (e.g. a4length).

Usage

The page dimensions of the report are read from the `4rp` file. This function is used to override the values found there.

For BDL ASCII reports being run in graphical mode (compatibility reports, no `4rp` file is being used), this function is used to set values for the page height and page width.

See [Dimension Resolver](#) on page 801 for additional examples of the strings that can be used in these parameters.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configurePDFDevice`

Configure the PDF output.

Syntax

```
fgl_report_configurePDFDevice(
    fontDirectory STRING,
    antialiasFonts BOOLEAN,
    antialiasShapes BOOLEAN,
    monochrome BOOLEAN,
    fromPage INTEGER,
    toPage INTEGER)
```

1. *fontDirectory* - Absolute path to the directory containing the font files
2. *antialiasFonts* - Configures whether fonts should be rendered using antialiasing. The default value is false.
3. *antialiasShapes* - Configures whether shapes should be rendered using antialiasing. The default value is false.
4. *monochrome* - Configures whether color values should be converted to monochrome output. The default value is false.
5. *fromPage* - Selects the lower bound of the range of pages to include in the PDF document. The default value is 1.

6. *toPage* - Selects the upper bound of the range of pages to include in the PDF document. By default all pages are included.

Usage

Function to configure the PDF output.

This function is applicable when PDF output has been selected by a call to the function [fgl_report_selectDevice](#). All arguments to this function are optional (indicated by passing a null value). If the PDF document should be written to a file, the general function [fgl_report_setOutputFileName](#) and [fgl_report_selectPreview](#) are available for this purpose.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configurePDFFontEmbedding`
Configure the font embedding in PDF output.

Syntax

```
fgl_report_configurePDFFontEmbedding(  
    preferUnicodeEncoding BOOLEAN )
```

1. *preferUnicodeEncoding* - Specified to encode characters in UNICODE whenever possible. This option should be set if non-Latin characters are used in the document. Unsetting the option improves processing speed and yields slightly smaller documents. By default the parameter has a value of TRUE.

Usage

This function configures the font embedding in PDF output.

This function is applicable when PDF output has been selected by a call to the function [fgl_report_selectDevice](#).

The argument to this function is optional, indicated by passing a null value.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureRTFDevice`
Configure RTF output for Microsoft™.

Syntax

```
fgl_report_configureRTFDevice(  
    fromPage INTEGER,  
    toPage INTEGER,  
    imagesResolution INTEGER,  
    imagesFormat STRING )
```

1. *fromPage* - Selects the lower bound of the range of pages to include in the RTF document. The default value is 1.
2. *toPage* - Selects the upper bound of the range of pages to include in the RTF document. Per default all pages are included.
3. *imagesResolution* - Specifies the resolution of embedded images. In addition to ImageBoxes, content from BarCodeBoxes, business charts and HTMLBoxes are embedded as images.
4. *imagesFormat* - One of : "png", "jpg". Controls the format of images embedded in the RTF document. Select jpg for compactness, png for lossless compression.

Usage

Function to configure RTF output.

This function is applicable when RTF output has been selected by a call to the function [fgl_report_selectDevice](#). All arguments to this function are optional (indicated by passing a null value).

If the RTF document should be written to a file, the general functions [fgl_report_setOutputFileName](#) and [fgl_report_selectPreview](#) are available for this purpose.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureSVGDevice`
Configure the SVG output.

Syntax

```
fgl_report_configureSVGDevice(
    antialiasFonts BOOLEAN,
    antialiasShapes BOOLEAN,
    embedFonts BOOLEAN,
    charsetToEmbed STRING )
```

1. *antialiasFonts* - Configures whether fonts should be rendered using antialiasing. The default value is false.
2. *antialiasShapes* - Configures whether shapes should be rendered using antialiasing. The default value is false.
3. *embedFonts* - Controls whether fonts are embedded within the document. The default value is true.
4. *charsetToEmbed* - SVG offers the possibility to embed fonts within the document. Documents with embedded fonts are slightly larger, but they offer the advantage of exact reproduction. Normally one would embed exactly those characters that were used throughout the document, but that would require reading the entire document before creating any SVG output. This attribute provides a solution that does not compromise streaming. The characters in the specified character set are embedded, not requiring all of them to have been used. Note that the character sets are neither restricted to 255 characters, nor is there any restriction on which unicode characters are used. Furthermore, an entry in the character set can be a sequence of unicode characters, thus allowing for ligatures. Valid values are:
 - **DEFAULT** - embeds all characters from the code pages iso-8859-1 through iso-8859-10 (about 600 characters).
 - **ISO-8859-15** - embeds all characters from the named code page (about 200 characters).
 - **ALL** - embeds all characters available in the selected fonts. This option should only be used when Asian characters are needed, since the size of the glyph definitions will significantly grow the document size.

Usage

Function to configure the SVG output.

This function is applicable when SVG output has been selected by a call to the function [fgl_report_selectDevice](#). All arguments to this function are optional (indicated by passing a null value). If the SVG document should be written to a file, the general function [fgl_report_setOutputFileName](#) and [fgl_report_selectPreview](#) are available for this purpose.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureSVGPreview`

Select how the document is handled by the SVG previewer.

This function is available only for the device SVG when previewing is selected.

Syntax

```
fgl_report_configureSVGPreview(  
    preview STRING)
```

1. *preview* - There are four possible options:

Preview	Makes the previewer visible and shows the document in a tab folder.
ShowPrintDialog	Pops up the print dialog allowing the user to select and configure a printer. The document is printed in the background and the previewer main window is not shown.
PrintOnDefaultPrinter	Prints the report silently on the default printer. The previewer main window is not shown.
PrintOnNamedPrinter	Prints the report silently on the a named printer. The previewer main window is not shown. The printer is named by a call to fgl_report_setSVGPrinterName() . and the page range can be set by calling fgl_report_setSVGPageRange() . If the previewer is on Windows, then the paper source can be selected with the function fgl_report_setSVGPaperSource . Additional parameters that can be set for printing from the SVG previewer include fgl_report_setSVGCopies and fgl_report_setSVGSheetCollate .

Usage

This function is available only for the device SVG when previewing is selected.

Important: Preview options set by `fgl_report_configureSVGPreview` are only valid when previewing with Genero Report Viewer in a Desktop configuration (`fgl_report_selectDevice("SVG")`). They are not valid when previewing with Genero Report Viewer for HTML5 in a Web configuration (`fgl_report_selectDevice("Browser")`).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

```
fgl_report_configureXLSDevice  
Configure the XLS (Excel) output.
```

Syntax

```
fgl_report_configureXLSDevice(  
    fromPage INTEGER,  
    toPage INTEGER,  
    removeWhitespace INTEGER,  
    ignoreRowAlignment INTEGER,  
    ignoreColumnAlignment INTEGER,
```

```
removeBackgroundImages INTEGER,
mergePages INTEGER )
```

1. *fromPage* - Selects the lower bound of the range of pages to include in the XLS document. The default value is 1.
2. *toPage* - Selects the upper bound of the range of pages to include in the XLS document. By default all pages are included.
3. *removeWhitespace* - Controls whether or not cells should be created for empty strings. By default whitespace is stripped from the document.
4. *ignoreRowAlignment* - When set, only those objects that are entirely above or entirely below each other will go in separate rows. When set, the option reduces the amount of rows thereby losing the horizontal alignment. The placement is not changed so that stacked items remain stacked. By default row alignment is ignored.
5. *ignoreColumnAlignment* - When set, only those objects that are entirely to the left or entirely to the right of each other will go in separate columns. When set, the option reduces the amount of columns thereby losing the vertical alignment. The placement is not changed so that adjacent items remain adjacent. By default column alignment is ignored.
6. *removeBackgroundImages* - Controls the behavior in case an IMAGEBOX is partially obscured by another element. When set, the image is removed from the resulting document otherwise the handling is as with any other case of overlapping items. By default, background images are removed.
7. *mergePages* - Controls the behavior when the report has more than one page. By default a separate sheet is created per page. Setting this parameter causes the pages to be merged, creating a single result sheet unless the sheet has more than 65536 rows; in that case, the exceeding rows spill over into extra sheets. Setting this parameter and using a standard page size is the recommended way to produce single-sheet output; using a huge custom page size instead can adversely affect memory reclamation and performance.

Usage

Function to configure the XLS (Excel) output.

This function is applicable when XLS output has been selected by a call to the function [fgl_report_selectDevice](#). All arguments to this function are optional, indicated by passing a null value. If the XLS document should be written to a file, the general functions [fgl_report_setOutputFileName](#) and [fgl_report_selectPreview](#) are available for this purpose.

See also [Sending Data to an Excel spreadsheet](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_configureXLSXDevice`

Configure Excel output in XLSX format.

Syntax

```
fgl_report_configureXLSXDevice (
  fromPage INTEGER,
  toPage INTEGER,
  removeWhitespace INTEGER,
  ignoreRowAlignment INTEGER,
  ignoreColumnAlignment INTEGER,
  removeBackgroundImages INTEGER,
  mergePages INTEGER)
```

1. *fromPage* - Selects the lower bound of the range of pages to include in the XLS document. The default value is 1.

2. *toPage* - Selects the upper bound of the range of pages to include in the XLS document. By default all pages are included.
3. *removeWhitespace* - Controls whether or not cells should be created for empty strings. By default whitespace is stripped from the document.
4. *ignoreRowAlignment* - When set, only those objects that are entirely above or entirely below each other will go in separate rows. When set, the option reduces the amount of rows thereby losing the horizontal alignment. The placement is not changed so that stacked items remain stacked. By default row alignment is ignored.
5. *ignoreColumnAlignment* - When set, only those objects that are entirely to the left or entirely to the right of each other will go in separate columns. When set, the option reduces the amount of columns thereby losing the vertical alignment. The placement is not changed so that adjacent items remain adjacent. By default column alignment is ignored.
6. *removeBackgroundImages* - Controls the behavior in case an IMAGEBOX is partially obscured by another element. When set, the image is removed from the resulting document otherwise the handling is as with any other case of overlapping items. By default, background images are removed.
7. *mergePages* - Controls the behavior when the report has more than one page.

Usage

Function to configure Excel output in XLSX format. The functionality is identical to the existing "XLS" output except for:

- The format can be opened only by newer versions of Microsoft™ Excel (beginning with 2007). It is possible to download a backward compatibility pack from Microsoft™ that will allow opening of these files with older versions. Beware that in this case worksheets containing more than 65536 rows will be truncated at this limit.
- The new format overcomes the 65536 row limit of the "XLS" format. This is the main motivation for introducing this format.
- The new format is generated with constant memory consumption so that very large documents can be produced without exhausting heap space.

This function is applicable when "XLSX" output has been selected by a call to the function [fgl_report_selectDevice](#). All arguments to this function are optional, indicated by passing a null value. If the XLSX document should be written to a file, the general functions [fgl_report_setOutputFileName](#) and [fgl_report_selectPreview](#) are available for this purpose.

See also [Sending Data to an Excel spreadsheet](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_createProcessLevelDataFile`

Configures the report execution to output an XML datafile.

Syntax

```
fgl_report_createProcessLevelDataFile(
    dataFileName STRING )
RETURNING driver om.SaxDocumentHandler
```

1. *dataFileName* - Name of the data file to generate.
2. *driver* - Returns an `om.SaxDocumentHandler` if no error occurred.

Usage

Function that configures the report execution to output an XML datafile.

This function configures the execution to output data to the specified file. Calling this function will configure the report engine to output the data file only. You can use the function [fgl_report_runReportFromProcessLevelDataFile](#) to run a report getting the data from this XML file.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_findResourcePath`

Returns the path to a resource searching first `FGLRESOURCEPATH`, then `FGLDBPATH` and finally `DBPATH`. (deprecated!)

Syntax

```
fgl_report_findResourcePath(
  reportName STRING )
```

1. *reportName* - Specify the name of the report.

Usage

The function returns the absolute path to the resource specified in the argument. In case that a file cannot be found the function returns the argument passed.

Important: The function is deprecated since it does not work in distributed mode. The function `fgl_report_loadCurrentSettings` now searches the path variables, making this function obsolete.

The function is typically used in conjunction with the function `fgl_report_loadCurrentSettings()` as in this example:

```
IF
  fgl_report_loadCurrentSettings(fgl_report_findResourcePath("OrderReport.4rp"))
THEN
  ...
```

Assuming `FGLRESOURCEPATH` is set to `/home/fjs/reports/lists:/home/fjs/correspondence` and the report `OrderReport.4rp` is located at `/home/fjs/correspondence/OrderReport.4rp`, that report will be loaded.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_runReportFromProcessLevelDataFile`

Runs a report from a process level file.

Syntax

```
fgl_report_runReportFromProcessLevelDataFile(
  out om.SaxDocumentHandler,
  fileName STRING )
RETURNING ok INTEGER
```

1. *out* - A output pipe obtained by a call to [fgl_report_commitCurrentSettings](#) or a similar function.
2. *fileName* - A file name of an XML file previously created by a call to [fgl_report_createProcessLevelDataFile](#).
3. *ok* INTEGER - "true" if no error occurred.

Usage

Function that runs a report from a process level file, previously created by a call to [fgl_report_createProcessLevelDataFile](#).

This function will replay the report from the file thereby replacing the running of the report (`START REPORT`, `OUTPUT TO REPORT`, `FINISH REPORT` statements).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_runFromXML`

Replays the report from an XML file.

Syntax

```
fgl_report_runFromXML(  
    dataFileName STRING)  
RETURNING ok INTEGER
```

1. *dataFileName* - file name of an XML file containing the data.
2. *ok* - "true" if no error occurred.

Usage

Function that can run a report using as the data source an arbitrary XML file. This would be a file that was not created by a Genero program using [fgl_report_createProcessLevelDataFile](#).

This function will replay the report from the file, replacing the running of the report (the `START REPORT`, `OUTPUT TO REPORT`, and `FINISH REPORT` statements).

The function sets up the report engine based on the current settings that have previously been loaded by a call to [fgl_report_loadCurrentSettings](#), and may have been modified by calls to [fgl_report_selectDevice](#) or [fgl_report_selectPreview](#). The function automatically calls [fgl_report_commitCurrentSettings](#).

See [Data From an XML File](#) for an example program.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

The Report Design

When it comes to the report design document (`.4rp`), you must declare the XSD associated with the XML file, as no data schema (`.rdd`) is created, as reports using this function do not include the Genero BDL report statements. See [Support for arbitrary XML data sources](#) on page 667.

`fgl_report_selectDevice`

Select the output device.

Parameters

See [Usage](#).

Syntax

```
fgl_report_selectDevice(  
    device STRING )
```

1. *device* STRING - Sets the output device from this list:

- HTML
- Image
- PDF
- Postscript
- Printer - Setting this parameter value selects server-side silent printing. Report Writer selects the printer that best meets the criteria specified by the functions listed in [the Printer Functions section](#).
- RTF (Microsoft™ RTF)
- OORTF (Open Office RTF)
- SVG
- XLS
- XLSX
- Browser

Usage

Function to select the output device.

Selecting a different output device changes the current settings. The choices are listed in Parameters.

Device-specific configuration functions are also available, as shown in [Table 163: Device-specific functions](#) on page 611.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setEnvironment`

Specifies variable values in the private environment of the report.

Syntax

```
fgl_report_setEnvironment(
  values om.SaxAttributes )
```

1. *values* - Attribute set containing the values.

Usage

Can be used to specify the value of environment variables like *GREOUTPUTDIR* or user-defined variables, for the purpose of using the values in calls to the RTL function `Runtime.getEnvironmentVariable()`.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_selectLogicalPageMapping`

Configures the mapping of logical pages to physical pages.

Syntax

```
fgl_report_selectLogicalPageMapping(
  mapping STRING )
```

1. *mapping* - one of oneToOne, labels, or multipage

Usage

Function that configures the mapping from logical pages (page dimensions in the 4r_p file) to physical pages (actual page dimensions on the device).

Table 164: Mapping options

Mapping Option	Description
oneToOne	The logical and physical pages are identical. Default value.
labels	For label printing it is advisable to design one label rather than creating a fixed $n \times m$ layout. Such a layout can be printed on an arbitrary physical $n \times m$ layout as long as the physical labels have at least the size of the logical labels. The physical dimensions of the layout must be configured using the function fgl_report_configureLabelOutput and fgl_report_setPaperMargins . The margins within a label can be configured by calling fgl_report_setPageMargins .
multipage	For ISO and JIS sized pages it is possible to print several pages per page. This is achieved by setting the mapping to this value and calling the function fgl_report_configureMultipageOutput to configure the required number of pages.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_selectPreview

The `fgl_report_selectPreview` function determines whether the report is shown in the previewer or written to a file on disk.

Syntax

```
fgl_report_selectPreview(
  preview INTEGER )
```

1. `preview` should be set to 0 (FALSE) or 1 (TRUE). When set to FALSE, the file is generated on disk rather than being previewed.

Usage

Function to select preview mode.

This function sets the output to be shown in the previewer. Suitable to all output formats except "Image". The appropriate software needs to be installed on the client:

- For PDF: Acrobat Reader (or any other PDF reader)
- For SVG: the Genero Report Viewer is a part of the Genero Desktop Client
- For RTF: Microsoft™ Word
- For XLS and XLSX: Microsoft™ Excel
- For OORTF: Open Office (or any free Office), Open Office Writer is sufficient
- For HTML and Browser: Any browser

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_setAutoformatType

Sets the auto-formatting type if no 4rp template is specified.

Syntax

```
fgl_report_setAutoformatType(  
    type STRING )
```

1. *type* - One of "COMPATIBILITY", "FLAT LIST"

Usage

Function that sets the auto-formatting type if no 4rp template is specified. By default such a report would be rendered in [compatibility mode](#); this function provides additional rendering options:

- A compatibility report - the report will be output in ASCII format, as in [legacy reports](#)
- A flat list - this is a simple list design, similar to the output from the List Report template available from the Genero Studio main menu option **File >> New, Reports, Report Designs**. This design is particularly well suited to produce Excel output from arbitrary reports.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_setBrowserDocumentDirectory

Configures the document directory for browser viewing.

Syntax

```
fgl_report_setBrowserDocumentDirectory(  
    directory STRING)
```

1. *directory* - The directory to hold the report files for use by Genero Report Viewer for HTML5, when Browser is selected as the output device.

Usage

See [Send report data to a browser](#) on page 595.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_setBrowserDocumentDirectoryURL

Configures the URL by which the document directory for browser viewing will be visible from a web server.

Syntax

```
fgl_report_setBrowserDocumentDirectoryURL(  
    directory STRING)
```

1. *directory* - The URL by which the document directory for browser viewing will be visible from a web server.

Usage

For use by Genero Report Viewer for HTML5, when Browser is selected as the output device. See [Send report data to a browser](#) on page 595.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setBrowserFontDirectory`

Configures the font directory for browser viewing.

Syntax

```
fgl_report_setBrowserFontDirectory(
    directory STRING)
```

1. *directory* - The directory to hold the font files for use by Genero Report Viewer for HTML5, when Browser is selected as the output device.

Usage

See [Send report data to a browser](#) on page 595.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setBrowserFontDirectoryURL`

Configures the URL by which the font directory for browser viewing will be visible from a web server.

Syntax

```
fgl_report_setBrowserFontDirectoryURL(
    directory STRING)
```

1. *directory* - The URL by which the font directory for browser viewing will be visible from a web server.

Usage

For use by Genero Report Viewer for HTML5, when Browser is selected as the output device. See [Send report data to a browser](#) on page 595.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setCallbackLocalization`

Configure the report to use a function to retrieve localized field titles.

Syntax

```
fgl_report_setCallbackLocalization(
    share BOOLEAN )
```

1. *share* - when set, the report engine will call the GRE library function `report_getFieldCaption` for each field, to retrieve the field caption.

Usage

By default the field titles in a report are retrieved from Genero localization files compiled by the command `fglmkstr`. This function configures the report engine to call the function `report_getFieldCaption(matchName, fieldName)` instead, and doesn't use the localization file.

See [Localized strings](#) on page 596.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setDistributedRequestingUserName`

Specify end user's name for the purpose of identifying log entries in the case of distributed processing.

Syntax

```
fgl_report_setDistributedRequestingUserName (
    requestingUserName STRING )
```

1. *requestingUserName* is the user name.

Usage

In order to distinguish between log entries originating from different users, the messages can be prefixed with the "requestingUserName" value. The requesting user name is an arbitrary string defined by the client, where you can set the name of the end user who submitted the job (for example). The default is "not set".

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setImageShrinkImagesToPageContent`

Configure image cropping.

Syntax

```
fgl_report_setImageShrinkImagesToPageContent (
    value BOOLEAN )
```

1. *value* - true or false.

Usage

Function to configure image cropping.

Sets whether the images produced are cropped to the page content (size of the page box) or have full page size.

The default value is false.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setImageUsePageNamesAsFileNames`

Configure image file name generation.

Syntax

```
fgl_report_setImageUsePageNamesAsFileNames (
    value BOOLEAN )
```

1. *value* - true or false.

Usage

Function to configure image file name generation.

Sets whether the page names ("name" attribute) in the document should be used as image file names. In the case that a name is not unique, a disambiguation number is appended. In the case that a page does not set the "name" attribute, the default naming scheme explained in [fgl_report_configureImageDevice](#) on page 615 is used.

The default value is false.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setOutputFileName`

Configure the file location for the device output. This function works for all output formats except Image.

Syntax

```
fgl_report_setOutputFileName(
    fileName STRING )
```

1. *fileName* - Enter a file name, with or without a file suffix, or a complete path with a file name.

If no suffix is specified, a device-specific suffix is appended, with the extension corresponding to the output type chosen by the [fgl_report_selectDevice\(\)](#) API call.

If you do not specify a path, the file is written to the current directory.

Usage

Function to configure the file location for the device output. This function works for all output formats except for the Image output format. For the Image output format, you must call the function [fgl_report_configureImageDevice](#).

You can use this function when file preview has been disabled, by a call to [fgl_report_selectPreview\(FALSE\)](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPageMargins`

Configure the logical margins of a report.

Syntax

```
fgl_report_setPageMargins(
    topMargin STRING,
    bottomMargin STRING,
    leftMargin STRING,
    rightMargin STRING)
```

1. *topMargin* - top margin value of the logical page (e.g. 0.5cm).
2. *bottomMargin* - bottom margin value of the logical page (e.g. 0.5cm).
3. *leftMargin* - left margin value of the logical page (e.g. 0.5cm).
4. *rightMargin* - right margin value of the logical page (e.g. 0.5cm).

Usage

Function that configures the logical margins of a report.

The logical margins of the report are read from the `4rp` file. This function is used to override the values found there.

For BDL ASCII reports (compatibility reports, no `4rp` file is specified), this function is used to set the values of the parameters:

For additional information about the units that can be used to specify the margin values, see [Dimension Resolver](#).

In the case of label printing (see [fgl_report_selectLogicalPageMapping](#)), this function specifies the margins within a label. Similarly, when multi page output is selected (see [fgl_report_selectLogicalPageMapping](#)) the values specify the margins of the logical pages which can be smaller than the physical margins since the required width is not limited by device limitations but by aesthetic aspects only.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPaperMargins`

Configure the physical margins of a report.

Syntax

```
fgl_report_setPaperMargins(
    topMargin STRING,
    bottomMargin STRING,
    leftMargin STRING,
    rightMargin STRING )
```

1. *topMargin* - top margin value of the physical page (e.g. 0.5cm).
2. *bottomMargin* - bottom margin value of the physical page (e.g. 0.5cm).
3. *leftMargin* - left margin value of the physical page (e.g. 0.5cm).
4. *rightMargin* - right margin value of the physical page (e.g. 0.5cm).

Usage

Function that configures the physical margins of a report

The physical margins of the report can be set by this function for the case that either label printing or multi page output has been selected by a call to [fgl_report_selectLogicalPageMapping](#).

For additional information about the units that can be used to specify the margin values, see [Dimension Resolver](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPageSwappingThreshold`

Sets the threshold for page-to-disk swapping.

Syntax

```
fgl_report_setPageSwappingThreshold(
    value INTEGER )
```

1. *value* - A positive integer greater than zero that specifies the maximum number of pages that may be held in main memory.

Usage

Function that sets the threshold for page-to-disk swapping to prevent memory exhaustion on very large documents that use " [Page N of M](#)"

The function specifies the maximum number of pages that may be held in main memory. When the value is exceeded, pages are swapped to the disk. This parameter is only needed for very large report that contain references to the total number of pages. alternatively it is possible to grow the amount of memory that the Java™ JVM may allocate by setting the parameter -Xmx (e.g. "java -Xmx512m" for 512 MB) in the script `$GREDIR/bin/greportwriter[.bat]`. By default the value is not set so that pages are never swapped to disk.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPDFJPEGImageEncoding`

Configure the encoding method of embedded images in PDF output.

Syntax

```
fgl_report_setPDFJPEGImageEncoding(
    encodeImagesAsJPEG STRING,
    jpegQuality BOOLEAN )
```

1. *encodeImagesAsJPEG* - Specifies images to be encoded in JPEG format. The default value is False.
2. *jpegQuality* - Sets the compression quality to a value between 0 and 1. By default a value of 1 is used (highest quality).

Usage

Function to configure the encoding method of embedded images in PDF output.

Optional JPEG image encoding can be configured to shrink the size of PDF documents using this function. This function is applicable when PDF output has been selected by a call to the function [fgl_report_selectDevice](#). All arguments to this function are optional (indicated by passing a null value).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPDFImageResolution`

Configure the resolution of embedded images in PDF output.

Syntax

```
fgl_report_setPDFImageResolution(
    imagesResolution INTEGER )
```

1. *imagesResolution* - Specifies the maximum resolution of embedded images.

Usage

Function to configure the resolution of embedded images in PDF output.

This function is applicable when PDF output has been selected by a call to the function [fgl_report_selectDevice](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterChromaticity`

Control color selection of the printer.

Syntax

```
fgl_report_setPrinterChromaticity(
    chromaticity STRING )
```

1. *chromaticity* - one of: monochrome|color.

Usage

Function to control color selection of the printer

Controls how the print data should be generated or processed. It does not confine the printer selection to printers with the specified capability. Default value is color. Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterCopies`

Specify the number of copies to be printed.

Syntax

```
fgl_report_setPrinterCopies(
    copies INTEGER )
```

1. *copies* is the number of copies to print.

Usage

Function to specify the number of copies to be printed.

Specifies the number of copies to be printed. Default value: 1

Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterDestinationUrl`

Specify an alternate destination for the spooled printer formatted data.

Syntax

```
fgl_report_setPrinterDestinationUrl(
    destination STRING )
```

1. *destination* - the destination URL.

Usage

Function to specify an alternate destination for the spooled printer formatted data.

A URL indicating an alternate destination for the spooled printer formatted data. Some print services will not support the notion of a destination other than the printer device and so will not support this attribute. A common use for this attribute will be applications which want to redirect output to a local disk file: e.g. "file:out.prn". A more platform independent way is to set the `fgl_report_setPrinterName` to "stdout" so that postscript is written to stdout or to specify the `fgl_report_setPrinterwriteToFile` attribute which causes postscript to be written to the specified file. Another alternative may be the use of the `fgl_report_selectDevice` to generate a PDF file .

By default, it is not set. Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterFidelity`

Select printer high fidelity mode.

Syntax

```
fgl_report_setPrinterFidelity(  
    fidelity INTEGER )
```

1. *fidelity* - true or false

Usage

Function to select printer high fidelity mode.

When set, all attributes of the printer have to meet the requested values, otherwise the printout will fail. When set to false, a reasonable attempt to print the document is acceptable. Default value is false. Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterJobImpressions`

Specify the total size in number of impressions.

Syntax

```
fgl_report_setPrinterJobImpressions(  
    jobImpressions INTEGER )
```

1. *jobImpressions* - number of jobs.

Usage

Function to specify the total size in number of impressions.

Specifies the total size in number of impressions of the document. An "impression" is the image (possibly many print stream pages in different configurations) imposed onto a single media page. The *jobImpressions* attribute describes the size of the job. This attribute is not intended to be a counter; it is intended to be useful routing and scheduling information. The printer may try to compute the attribute's value if it is not supplied. Even if a value is supplied, the printer may choose to change the value if the printer is able to compute a value that is more accurate than the supplied value. The printer may be able to determine the correct value for the *jobImpressions* attribute right at job submission or at a later time. Unlike

the [fgl_report_setPrinterJobMediaSheets](#) function, the value must not include the multiplicative factors contributed by the number of copies, specified by the [fgl_report_setPrinterCopies](#) function. Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterJobMediaSheets`

Specify the total number of media sheets.

Syntax

```
fgl_report_setPrinterJobMediaSheets (
    jobMediaSheets INTEGER )
```

1. *jobMediaSheets* INTEGER - number of sheets

Usage

Function to specify the total number of media sheets.

Specifies the total number of media sheets to be produced for this job. The *jobMediaSheets* attribute describes the size of the job. This attribute is not intended to be a counter; it is intended to be useful routing and scheduling information. The printer may try to compute the attribute's value if it is not supplied. Even if a value is supplied, the printer may choose to change the value if the printer is able to compute a value that is more accurate than the supplied value. The printer may be able to determine the correct value for the *jobMediaSheets* attribute right at job submission or at a later time. The value must include the multiplicative factors contributed by the number of copies, specified by the [fgl_report_setPrinterCopies](#) function.

By default, this is not set. Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterJobName`

Specify a name for the job.

Syntax

```
fgl_report_setPrinterJobName (
    jobName STRING )
```

1. *jobName* - name of the job.

Usage

Function to specify a name for the job.

Name of the print job useful for tracking the job. The value does not have to be unique. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterJobPriority`

Specify a priority for the job.

Syntax

```
fgl_report_setPrinterJobPriority(  
    jobPriority INTEGER)
```

1. *jobPriority* - priority of the job. Fails if the argument is not between 1 and 100.

Usage

Function to specify a priority for the job.

If supplied, the value specifies a priority for scheduling the job. A higher value specifies a higher priority. The value 1 indicates the lowest possible priority. The value 100 indicates the highest possible priority. Among those jobs that are ready to print, a printer must print all jobs with a priority value of n before printing those with a priority value of n-1 for all n. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_setPrinterJobSheets
Control job sheet printing.

Syntax

```
fgl_report_setPrinterJobSheets(  
    jobSheets STRING)
```

1. *jobSheets* - one of: none|standard

Usage

Function to control job sheet printing.

Controls if job start and end sheets are to be printed. Default: none. Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

fgl_report_setPrinterMediaName
Select the type of media to use.

Syntax

```
fgl_report_setPrinterMediaName(  
    mediaTray STRING)
```

1. *mediaTray* - one of: iso-a4-transparent|iso-a4-white|na-letter-transparent

Usage

Function to select the type of media to use.

Controls the type of media to choose. This function and the functions [fgl_report_setPrinterMediaSizeName](#) and [fgl_report_setPrinterMediaTray](#) are mutually exclusive. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterMediaSizeName`

Select the media size to be used.

Syntax

```
fgl_report_setPrinterMediaSizeName(  
    mediaSizeName STRING )
```

1. *mediaSizeName* - one of: a|b|c|d|e|executive|folio|invoice |iso-a0|iso-a1|iso-a2|iso-a3|iso-a4|iso-a5|iso-a6|iso-a7|iso-a8|iso-a10|iso-b0 |iso-b1|iso-b2|iso-b3|iso-b4|iso-b5|iso-b6|iso-b7|iso-b8|iso-b10|iso-c0|iso-c1 |iso-c2|iso-c3|iso-c4|iso-c5|iso-c6|iso-designated-long|iso-italian-envelope |iso-oufuko-postcard|jis-b0|jis-b1|jis-b2|jis-b3|jis-b4|jis-b5|jis-b6|jis-a7 |jis-b8|jis-b10|ledger|monarch-envelope|na-10x13-envelope|na-10x14-envelope |na-10x15-envelope|na-5x7|na-6x9-envelope|na-8x10|na-9x11-envelope |na-9x12-envelope|na-legal|na-letter|na-number-10-envelope|na-number-11-envelope |na-number-12-envelope|na-number-14-envelope|na-number-9-envelope |personal-envelope|quarto|tabloid

Usage

Function to select the media size to be used.

Selects the media size to be used. Normally this does not need to be specified, as the program will automatically select the smallest media onto which the current document can be printed without clipping or scaling. This function and the functions [fgl_report_setPrinterMediaTray](#) and [fgl_report_setPrinterMediaName](#) are mutually exclusive. Default value not set (is automatically selected). Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterMediaTray`

Select the tray of the printer.

Syntax

```
fgl_report_setPrinterMediaTray(  
    mediaTray STRING)
```

1. *mediaTray* STRING - one of: bottom|envelope|large-capacity|main|manual|middle|side|top

Usage

Function to select the tray of the printer.

Controls what tray to take the media from. This function and the functions [fgl_report_setPrinterMediaTray](#) and [fgl_report_setPrinterMediaName](#) are mutually exclusive. Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterName`

Select a specific printer by name.

Syntax

```
fgl_report_setPrinterName(  
    printerName STRING)
```

1. *printerName* - the name of the printer.

Usage

Function to select a specific printer by name.

Name of requested printer listed as "SERVICE". A special meaning is attached to the name "stdout". If this value is specified, then postscript is written to stdout. Default value is "Not Set". Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterNumberUp`

Specify the number of print stream pages for a single side of an instance.

Syntax

```
fgl_report_setPrinterNumberUp(  
    numberUp INTEGER )
```

1. *numberUp* - number of pages

Usage

Function to specify the number of print stream pages for a single side of an instance.

Specifies the number of print stream pages to impose upon a single side of an instance of selected medium. That is, if the *numberUp* value is *n*, the printer must place *n* print stream pages on a single side of an instance of the selected medium. To accomplish this, the printer may add some sort of translation, scaling or rotation.

Note: Since this feature is available only on a few printers, it is advisable to perform the necessary transformations using this API Default: "not set" Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterOrientationRequested`

Control the paper orientation.

Syntax

```
fgl_report_setPrinterOrientationRequested(  
    orientationRequested STRING)
```


1. *orientationRequested* - one of: landscape|portrait|reverse-landscape|reverse-portrait

Usage

Function to control the paper orientation.

Controls the paper orientation. Normally this value should not be set. The style sets the value internally by analyzing the page's width and height values. Default: not set Setting this option reduces the set of usable printers to those matching it.

Fails if argument is not one of the specified values.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterPageRanges`
Specify the ranges of pages to print.

Syntax

```
fgl_report_setPrinterPageRanges (
  pageRanges STRING )
```

1. *pageRanges* - page ranges following the syntax [1-9][0-9]*-[1-9][0-9]*[, [1-9][0-9]*-[1-9][0-9]*]*

Usage

Function to specify the ranges of pages to print.

Specifies the ranges of print stream pages that the printer uses for each copy of the document printed. Nothing is printed for any pages identified that do not exist in the document. Default: "not set" (everything is printed) Setting this option reduces the set of usable printers to those matching it.

Fails if argument does not follow syntax.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterPrintQuality`
Control the quality used by the printer.

Syntax

```
fgl_report_setPrinterPrintQuality (
  printQuality STRING )
```

1. *printQuality* - one of: draft|high|normal

Usage

Function to control the quality used by the printer

Specifies the print quality used by the printer. Default: not set Setting this option reduces the set of usable printers to those matching it.

Fails if argument is not one of the specified values

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterRequestingUserName`

Specify end user's name.

Syntax

```
fgl_report_setPrinterRequestingUserName(
    requestingUserName STRING )
```

1. *requestingUserName* - the user name.

Usage

Function to specify end user's name.

Specifies the end user's name who submitted the print job. A requesting user name is an arbitrary string defined by the client. The printer does not put the client-specific `requestingUserName` attribute into the print job's attribute set; rather, the printer puts in a `jobOriginatingUserName` attribute. This means that services which support specifying a username with this attribute should also report a `jobOriginatingUserName` in the job's attribute set. Note that many print services may have a way to independently authenticate the user name, and so may state support for a requesting user name, but in practice will then report the user name authenticated by the service rather than that specified by this attribute. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterResolution`

Specify an exact resolution for the printer.

Syntax

```
fgl_report_setPrinterResolution(
    resolution INTEGER )
```

1. *resolution* - the resolution of the printer. You can set a resolution for both the X-axis and the Y-axis, or you can set the resolution for the X-axis and the Y-axis separately. You can specify DPI (dots per inch) or DPCM (dots per centimeter). If not specified, DPI is the default measure.

Table 165: Valid resolution entries

Value	Description (X-axis, Y-axis, Measure)
300	300,300,dpi
300 dpcm	300,300,dpcm
300 , 150	300,150,dpi
300 , 150 , dpi	300,150,dpi
300 , 150 , dpcm	300,150,dpcm

Usage

Function to specify an exact resolution.

Specifies an exact resolution supported by a printer or to be used for the job. This attribute assumes that printers have a small set of device resolutions at which they can operate rather than a continuum.

By default, a resolution is not specified.

Setting this option reduces the set of usable printers to those matching it.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterSheetCollate`

Controls the collation of multiple copies.

Syntax

```
fgl_report_setPrinterSheetCollate(
    sheetCollate STRING)
```

1. *sheetCollate* - Can be either `collated` or `uncollated` (default)

Usage

Specifies whether or not the pages of the document are to be in sequence, when multiple copies of the document are specified by the `fgl_report_setPrinterCopies` function. When *sheetCollate* is `"collated"`, each copy of the document is printed with the pages in sequence. When *sheetCollate* is `"uncollated"`, each page is printed a number of times equal to the value of the `fgl_report_setPrinterCopies` attribute in succession.

For example, suppose a document produces two pages as output, `fgl_report_setPrinterCopies` is 6, and *sheetCollate* is `"uncollated"`; in this case six copies of the first page are printed, followed by six copies of the second page.

Tip: It is discouraged to set *sheetCollate* to `"collated"` since it requires caching the entire document which is undesirable for large documents. If necessary, produce the document the required number of times.

Setting this option reduces the set of usable printers to those who are able to perform the collation as requested.

The function fails if the argument is not one of the specified values.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterSides`

Specify the mapping of pages on the physical media.

Syntax

```
fgl_report_setPrinterSides(
    sides STRING)
```

1. *sides* - One of: `one-sided|two-sided-short-edge|two-sided-long-edge`

Usage

Function to specify the mapping of pages on the physical media.

Specifies how print-stream pages are to be imposed upon the sides of an instance of a selected medium, i.e., an impression.

- `one-sided`: Imposes each consecutive print-stream page upon the same side of consecutive media sheets.

- **two-sided-short-edge:** Imposes each consecutive pair of print stream pages upon front and back sides of consecutive media sheets, such that the orientation of each pair of print stream pages on the medium would be correct as if for binding along the short edge.
- **two-sided-long-edge:** Imposes each consecutive pair of print stream pages upon front and back sides of consecutive media sheets, such that the orientation of each pair of print stream pages on the medium would be correct as if for binding along the long edge.

By default, this is not set. Setting this option reduces the set of usable printers to those matching it.

Fails if the argument is not one of the specified values.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setPrinterWriteToFile`

Specify a file where the report is written in postscript. (deprecated!)

Syntax

```
fgl_report_setPrinterWriteToFile(
    file STRING)
```

1. *file* - the destination file.

Usage

Function to specify a file where the report is written in postscript.

Important: The function is deprecated and replaced by the "Postscript" output format in calls to the function `fgl_report_selectDevice` in conjunction with the standard output specification function `fgl_report_setOutputFileName`.

A file name specifying a location where the report output is written in postscript format. If this attribute is set, all other IPP attribute values are ignored. Default: "not set" Setting this option reduces the set of usable printers to those matching it.

The function fails if *file* cannot be opened for writing.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setProcessLevelDataFile`

Configures the report execution to output an XML datafile in addition to the regular processing.

Syntax

```
fgl_report_setProcessLevelDataFile(
    dataFileName STRING )
```

1. *dataFileName* - Name of the data file to generate.

Usage

Function that configures the report execution to output an XML datafile in addition to the regular processing

Like the function `fgl_report_createProcessLevelDataFile`, this function causes a data file to be produced that can be used for archiving and/or reformatting, using the function `fgl_report_runReportFromProcessLevelDataFile`. However, this function causes the file to be created in addition to the regular processing; for example, a PDF file and a data file can be created at the

same time. Or, you could create a completely different layout for one of the files by specifying a different 4rp template.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setRTFMemoryThreshold`
Set the RTF memory threshold.

Syntax

```
fgl_report_setRTFMemoryThreshold (
    memoryThreshold INTEGER)
```

1. *memoryThreshold* - The threshold in bytes above which documents are swapped to disk.

Usage

Function to set the RTF memory threshold. Applies to both RFT (Microsoft™) and OORFT (Open Office).

In order to prevent exhaustion of main memory when processing large documents, the processor can be instructed to swap parts of the document to a temporary disk file when the document size exceeds this threshold. The default value is set to 10% of the total available heap space. The default heap space is 64Mb.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setSharePortWithGDC`
Configures the report engine to use the same port as the Genero Desktop Client for previewing.

Syntax

```
fgl_report_setSharePortWithGDC(
    share BOOLEAN)
```

1. *share* - Boolean that, when set to true, specifies that the engine will use `FGLSERVER` as the client port.

Usage

Function that configures the report engine to use the same port as the Genero Desktop Client for previewing.

By default, the port is shared and the port value specified in the `FGLSERVER` environment variable is used.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setSVGCompression`
Configure SVG compression.

Syntax

```
fgl_report_setSVGCompression(
    compressOutput BOOLEAN)
```

1. *compressOutput* - Boolean specifying whether the output should be compressed. The default value is false.

Usage

Function to configure SVG compression.

This function is applicable when SVG output has been selected by a call to the function [fgl_report_selectDevice](#). When set, the function causes SVG to be written in compressed form, causing files and streams to be shrunk to about a tenth of their original size. This can benefit the overall performance on slow networks.

The Genero Report Viewer automatically detects if files or streams are compressed and decompresses them on the fly as necessary.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setSVGCopies`

Specify the number of copies to be printed.

Syntax

```
fgl_report_setSVGCopies(  
  copies INTEGER)
```

1. *copies* - the number of copies to print.

Usage

Specifies the number of copies to be printed. The default is one (1).

The function applies only if the option **PrintOnNamedPrinter** is chosen in a call to [fgl_report_configureSVGPreview](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setSVGOrientationRequested`

Function to control the paper orientation.

Syntax

```
fgl_report_setSVGOrientationRequested(  
  orientationRequested String)
```

1. *orientationRequested* - Valid values are "landscape" or "portrait".

Usage

This function can pre-select (in the case of printing via the print dialog) or set the value of the orientation property.

Normally this value should not be set. The value is set internally by analyzing the page's width and height values. If a page's height is greater than its width, the orientation is portrait, otherwise it is landscape.

However, some printers can be loaded with both portrait and landscape paper. Without the printing software being aware of what type is currently loaded, the wrong orientation setting can cause a misprint. In the case of matrix or label printers which can be loaded with paper of various dimensions (without the

printer or the driver being aware of the current format), the value should be set to "portrait" for landscape reports when the printer contains landscape paper.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setSVGPageRange`

Select which pages should be printed.

Syntax

```
fgl_report_setSVGPageRange(
    fromPage INTEGER,
    toPage INTEGER )
```

1. *fromPage* - Selects the lower bound of the range of pages to print. The default value is 1.
2. *toPage* - Selects the upper bound of the range of pages to create print. By default, all pages are printed.

Usage

Function to select which pages should be printed.

The function applies only if the option **PrintOnNamedPrinter** is chosen in a call to [fgl_report_configureSVGPreview](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setSVGPaperSource`

Select the input source of the printer.

Syntax

```
fgl_report_setSVGPaperSource(
    paperSource STRING)
```

1. *paperSource* - one of: Auto, Cassette, Envelope, EnvelopeManual, FormSource, LargeCapacity, LargeFormat, Lower, Middle, Manual, OnlyOne, Tractor, SmallForm

Usage

Function to select the input source of the printer. Controls what source to take the paper from.

The function applies only if the option **PrintOnNamedPrinter** is chosen in a call to [fgl_report_configureSVGPreview](#). The functionality is available only on Windows™.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setSVGPrinterName`

Select a specific printer by name.

Syntax

```
fgl_report_setSVGPrinterName(
    printerName STRING )
```

1. *printerName* is the name of the printer.

Usage

Function to select a specific printer by name.

The function applies only if the option **PrintOnNamedPrinter** is chosen in a call to [fgl_report_configureSVGPreview](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setSVGSheetCollate`

Control the collation of multiple copies.

Syntax

```
fgl_report_setSVGSheetCollate (
    sheetCollate STRING )
```

1. *sheetCollate* - one of collated|uncollated.

If *sheetCollate* is:

- **collated**, each copy of the document is printed with the pages in sequence.
- **uncollated**, each page is printed a number of times equal to the value of the `fgl_report_setSVGCopies` attribute in succession. This is the default.

Usage

Function to control the collation of multiple copies.

Specifies whether the pages of the document are to be in sequence, when multiple copies of the document are specified by the `fgl_report_setSVGCopies` function.

For example, if a document produces two pages as output, `fgl_report_setSVGCopies` is 6, and `sheetCollate` is "uncollated", then six copies of the first page are printed followed by six copies of the second page.

Note: We discourage setting `sheetCollate` to "collated" since it requires caching the entire document, which is undesirable for large documents. If necessary, produce the document the required number of times.

This function applies only if the option **PrintOnNamedPrinter** is chosen in a call to [fgl_report_configureSVGPreview](#).

This function fails if the argument is not one of the specified values.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setXLSMergeCells`

Configure cell merging in XLS (Excel) output.

Syntax

```
fgl_report_setXLSMergeCells (
    mergeCells BOOLEAN)
```


1. *mergeCells* - Controls the behavior when an item of the report occupies more than one cell. Setting this value causes the cells to be merged into one cell. The value is TRUE by default.

Usage

Function to configure cell merging in XLS (Excel) output.

This function is applicable when XLS output has been selected by a call to the function [fgl_report_selectDevice](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_setXLSXMergeCells`

Configure cell merging in XLSX (Excel) output.

Syntax

```
fgl_report_setXLSXMergeCells (
    mergeCells BOOLEAN)
```

1. *mergeCells* - Controls the behavior when an item of the report occupies more than one cell. Setting this value causes the cells to be merged into one cell. The value is TRUE by default.

Usage

Function to configure cell merging in XLSX (Excel) output.

This function is applicable when XLSX output has been selected by a call to the function [fgl_report_selectDevice](#).

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

`fgl_report_stopGraphicalCompatibilityMode`

Restore text-based report output.

Syntax

```
fgl_report_stopGraphicalCompatibilityMode ( )
```

Usage

Function to restore text-based report output.

The graphical compatibility mode switched on by calling [fgl_report_loadCurrentSettings\(NULL\)](#) is switched off using this function.

For a generic example of Genero code using a reporting function, see [Using report output functions](#) on page 610. This example may not use the specific function discussed in this topic, however it provides details on where you would place this (and other) report output functions.

Functions to set Report Metadata for Compatibility Reports
Use these functions to set report metadata for compatibility reports.

Table 166: Functions to set Report Metadata for Compatibility Reports

Function	Description
<code>fgl_report_setAuthor (author STRING)</code>	Function to set the report author metadata value.
<code>fgl_report_setTitle (title STRING)</code>	Function to set the report title metadata value.
<code>fgl_report_setCreator (creator STRING)</code>	Function to set the report creator metadata value.
<code>fgl_report_setSubject (subject STRING)</code>	Function to set the report subject metadata value.
<code>fgl_report_setKeywords (keywords STRING)</code>	Function to set the report keywords metadata value.

Important: These Report Metadata functions are intended to be used for compatibility reports only. To set the metadata for standard reports, select the root node of the report in the Report Designer Properties View and set the appropriate properties.

If the target file format supports meta data, then the value is inserted into the target document. Typically this is a whitespace-separated list of key words. Calling this function to set the value supersedes the value specified in the 4rp template.

Functions to introspect reports at runtime (librdd)
Use these functions to access the structures contained in the rdd files.

The module librdd (part of the standard reporting library libgre.42x) provides programmatic access to the structures contained in rdd files (files produced by the 4GL compiler from 4GL sources with the option `--build-rdd`).

Table 167: Functions to introspect reports at runtime

Function	Description
<code>rdd_getEveryRowFields (rddFileName STRING, reportName STRING) RETURNING fieldNames DYNAMIC ARRAY OF STRING</code>	Function that loads an rdd file and returns the field names of the fields contained in the first PRINT statement from the ON EVERY ROW section of the specified report.
<code>rdd_loadRddAndGetReportInfos (rddFileName STRING) RETURNING reportInfos DYNAMIC ARRAY OF rddReportInfo</code>	Function that loads a rdd file and returns a data structure describing the reports contained in this file.
<code>rdd_debugReportInfos (reportInfos DYNAMIC ARRAY of rddReportInfo)</code>	Function that displays the report information contained in a rdd file for debugging purposes.

The source of the module is located in `$GREDIR/src/api/librdd.4gl`.

`rdd_getEveryRowFields`

Function that loads an `rdd` file and returns the field names of the fields contained in the first `PRINT` statement from the `ON EVERY ROW` section of the specified report.

Syntax

```
rdd_getEveryRowFields (
    rddFileName STRING,
    reportName STRING)
RETURNING fieldNames DYNAMIC ARRAY OF STRING
```

1. `rddFileName` STRING - Full path to a `rdd` file.
2. `reportName` STRING - Name of the report as specified in the `4gl` source file.
3. Returns `fieldNames` DYNAMIC ARRAY OF STRING - Null if error occurred.

Usage

Function that loads an `rdd` file and returns the field names of the fields contained in the first `PRINT` statement from the `ON EVERY ROW` section of the specified report.

Files of type `rdd` files are produced from `4gl` source files by the 4GL compiler using the option `--build-rdd`.

`rdd_loadRddAndGetReportInfos`

Function that loads a `rdd` file and returns a data structure describing the reports contained in this file.

Syntax

```
rdd_loadRddAndGetReportInfos (
    rddFileName STRING)
RETURNING reportInfos DYNAMIC ARRAY OF rddReportInfo
```

1. `rddFileName` STRING - Full path to a `rdd` file.
2. Returns `reportInfos` DYNAMIC ARRAY OF `rddReportInfo` - Null if error occurred.

Usage

Function that loads a `rdd` file and returns a data structure describing the reports contained in this file.

Files of type `rdd` files are produced from `4gl` source files by the compiler using the option `--build-rdd`.

`rdd_debugReportInfos`

Function that displays the report information contained in a `rdd` file for debugging purposes.

Syntax

```
rdd_debugReportInfos (
    reportInfos DYNAMIC ARRAY OF rddReportInfo)
```

1. `reportInfos` DYNAMIC ARRAY OF `rddReportInfo` - Structure obtained by a call to `rdd_loadRddAndGetReportInfos`.

Usage

Function that displays the report information contained in a `rdd` file for debugging purposes.

The function displays debugging information for the data structure returned by a call to the function `rdd_loadRddAndGetReportInfos`.

Types used in librdd

These data types are used in the [librdd](#) module.

rddReportPrintElementInfo

```
TYPE rddReportPrintElementInfo RECORD
  unionType CHAR(1), #v=variable, e=expression, l=literal
  value STRING, #the value in case of type 'l'
  variableName STRING, #the variable name case of type 'v'
  variableType STRING #the variable or expression type in case of type 'v'
END RECORD
```

rddReportPrintInfo

```
TYPE rddReportPrintInfo RECORD
  controlPath DYNAMIC ARRAY OF rddTree, # Path into tree, e.g.
                                     # REPORT/ON EVERY ROW/IF/THEN/FOR,
                                     # REPORT/ON EVERY ROW/IF/ELSE/FOR
  printxName STRING,
  printElements DYNAMIC ARRAY OF rddReportPrintElementInfo
END RECORD
```

rddReportSectionInfo

```
TYPE rddReportSectionInfo RECORD
  name STRING, # FIRST PAGE HEADER, BEFORE GROUP customer_id, ON EVERY
  ROW, ...
  reportPrints DYNAMIC ARRAY OF rddReportPrintInfo
END RECORD
```

rddReportInfo

```
TYPE rddReportInfo RECORD
  reportName STRING,
  reportSections DYNAMIC ARRAY OF rddReportSectionInfo
END RECORD
```

rddTree

```
TYPE rddTree RECORD
  firstLine INTEGER,
  lastLine INTEGER,
  type CHAR(1), # l: loop: FOR, FOREACH, WHILE
                # c: conditional: CASE, IF, OTHERWISE, WHEN
                # i: interaction: MENU, {DISPLAY|INPUT} ARRAY, INPUT,
  PROMPT,
                #
                # ON {KEY|ACTION}, {BEFORE|AFTER} ..
                # d: declaration: GLOBALS, DEFINE, RECORD
                # o: other statements: TRY, CATCH
                # a: trigger in interaction or report
  fileId INTEGER, # references rddFiles (rddFile.id=fileId)
  fileIdEnd INTEGER, # references rddFiles (rddFile.id=fileIdEnd) =
                    # the file where the declaration ends
  name STRING # for example BEFORE GROUP
END RECORD
```

Functions to get error details

Use these functions to get the most current error details. The functions are provided as part of the `libgre.42x` library.

Table 168: Functions to return error details

Function	Description
<code>fgl_report_getErrorStatus()</code> RETURNING <i>ok</i> INTEGER	Returns the most recent error status.
<code>fgl_report_getErrorString()</code> RETURNING <i>message</i> STRING	Returns a human-readable error string.

`fgl_report_getErrorStatus`

Returns the most recent error status.

Syntax

```
fgl_report_getErrorStatus()  
RETURNING ok INTEGER
```

1. *ok* - TRUE if an error occurred, FALSE otherwise.

Usage

This function returns the most recent error status. The function should be called after calling `OUTPUT TO REPORT` and `FINISH REPORT`. Calling the function at any other point in time may return random results. The function returns TRUE in case of errors in GRE.

`fgl_report_getErrorString`

Returns a human-readable error string.

Syntax

```
fgl_report_getErrorString()  
RETURNING message STRING
```

1. *message* - An error description if an error occurred, an empty string otherwise.

Usage

This function returns the most recent error string. The function should be called after calling `OUTPUT TO REPORT` and `FINISH REPORT`. Calling the function at any other point in time may return random results. In case of an error, the function returns a human readable description of the error.

Pivot table library

The library `libpivot.4gl` provides functions to retrieve information about pivot tables contained in a `.4rp` file

- [Types defined in the pivot library](#) on page 654
- [Pivot library API](#) on page 654
- [pivot_debugPivotTables](#) on page 655
- [pivot_getLastErrorString](#) on page 655
- [pivot_load4rpAndGetPivotTables](#) on page 655

Types defined in the pivot library

Three types are defined for the pivot library: `pivotHierarchy`, `pivotMeasure`, and `pivotPivotTable`.

```

PUBLIC TYPE pivotHierarchy RECORD
  name STRING,
  title STRING,
  format STRING,
  value STRING,
  isNumeric STRING,
  enumValues STRING,
  computeTotal STRING,
  computeCount STRING,
  computeDistinctCount STRING,
  computeAverage STRING,
  computeMinimum STRING,
  computeMaximum STRING
END RECORD

PUBLIC TYPE pivotMeasure RECORD
  name STRING,
  title STRING,
  format STRING,
  value STRING,
  isNumeric STRING
END RECORD

PUBLIC TYPE pivotPivotTable RECORD
  name STRING, #PIVOTTABLE
  title STRING, #PIVOTTABLE
  drawAs STRING, #PIVOTTABLE
  computeAggregatesOnInnermostDimension STRING, #PIVOTTABLE
  displayRecurringValues STRING, #PIVOTTABLE
  hierarchiesDisplaySelection STRING, #PIVOTTABLE
  inputOrder STRING, #PIVOTTABLE
  displayFactRows STRING, #FACT
  measuresDisplaySelection STRING, #FACT
  outputOrder STRING, #FACT
  topN INTEGER, #FACT
  hierarchies DYNAMIC ARRAY OF pivotHierarchy,
  measures DYNAMIC ARRAY OF pivotMeasure
END RECORD

```

Pivot library API

Functions from the pivot table library.

Table 169: Pivot library functions

Function	Summary
FUNCTION <code>pivot_debugPivotTables</code> (<i>pivotTables</i> DYNAMIC ARRAY OF <code>pivotPivotTable</code>)	Displays the pivot table information contained in a 4rp file for debugging purposes
FUNCTION <code>pivot_getLastErrorString</code> () RETURNING <i>errorMessage</i> STRING	Retrieves the error string of the last function called that failed.
FUNCTION <code>pivot_load4rpAndGetPivotTables</code> (<i>fileName</i> String)	This function loads a 4rp file and returns a data structure describing

Function	Summary
RETURNING <i>status</i> BOOLEAN, <i>pivotTables</i> DYNAMIC ARRAY OF pivotPivotTable	the pivot tables in this file.

`pivot_debugPivotTables`

Displays the pivot table information contained in a 4rp file for debugging purposes

```
FUNCTION pivot_debugPivotTables(  
  pivotTables DYNAMIC ARRAY OF pivotPivotTable)
```

The function displays debugging information for the data structure returned by a call to the function `pivot_load4rpAndGetPivotTables`.

Parameters

- *pivotTables* DYNAMIC ARRAY OF pivotPivotTable - Structure obtained by a call to `pivot_load4rpAndGetPivotTables`

`pivot_getLastErrorString`

Retrieves the error string of the last function called that failed.

```
FUNCTION pivot_getLastErrorString()  
  RETURNING errorMessage STRING
```

Returns

- *errorMessage* STRING - null if no error occurred.

`pivot_load4rpAndGetPivotTables`

This function loads a 4rp file and returns a data structure describing the pivot tables in this file.

```
FUNCTION pivot_load4rpAndGetPivotTables(  
  fileName String )  
  RETURNING status BOOLEAN,  
  pivotTables DYNAMIC ARRAY OF pivotPivotTable
```

Parameters

- *fileName* STRING - Full path to a 4rp file

Returns

- *status* BOOLEAN - FALSE if an error occurred.
- *pivotTables* DYNAMIC ARRAY OF pivotPivotTable

Create the data schema

For each report program, a corresponding data schema must be created. The data schema then populates the Data View in the Genero Report Designer, providing a hierarchical representation of the data objects to be dragged and dropped into the report design.

Tools have been provided to allow for the generation of a data schema for an application.

Generate a data schema from a Genero BDL report program

After you write or modify a Genero report program, you must generate the data schema (`rdd`) file. This file is used by the Genero Report Designer to provide a list of data objects for use in the report design.

The data schema (`rdd`) file is based on the SQL statement in your Genero report application source file (`4gl`). This `rdd` file is used in the report design document (`4rd`) to populate the [Data View](#), providing details about the fields that will be streamed by the application. The schema contains the list of database columns that make up your data record, as well as grouping details.

Although the data for the report originally may have come from several different data tables, the PRINT statement in your BDL REPORT program block outputs the data as part of a single record. See the Genero Studio >> Report Writer documentation topic "Writing the BDL Program" for more information.

From the command line

Use the `--buildrdd` command-line option of the `fglcomp` tool to create a data schema (`rdd` file). For example:

```
fglcomp --build-rdd SimpleReport.4gl
```

The output of this command will be `SimpleReport.rdd`. The `rdd` file will be stored in the same location as the `4gl` file.

Note: If your Genero program contains multiple `4gl` files, run `fglcomp` against the file containing your REPORT program block.

From Genero Studio

Add `--build-rdd` to the **Compiler options** property for your Genero source (`4gl`) file to generate the `rdd` file automatically each time the `4gl` file is compiled. Select the `4gl` file listing in the application node of the Project tab to display its properties in the Properties View. The `rdd` file will be stored in the directory specified in the **Target Directory** property of the application node that contains the `4gl` file.

Create a report design document

The Genero Report Designer provides a graphical interface for designing your reports.

- [What's new in Genero Report Designer, v 3.00](#) on page 656
- [The Report Design Document](#) on page 657
- [Report Designer Reference](#) on page 722
- [RTL Class Reference](#) on page 806
- [Upgrading Genero Report Designer](#) on page 837

What's new in Genero Report Designer, v 3.00

This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication.

Table 170: Genero Report Designer, Version 3.00

Overview	Reference
Genero Report Designer provides a LastPageFooter section property.	See section (Section) on page 754.
Support of Intelligent Mail bar code type.	See intelligent-mail on page 795.

Overview	Reference
New <code>smartParse</code> bar code property for bar code Code-128. When enabled, this allows you to enter the bar code value, and the internal code will be computed for you resulting in the shortest visual representation.	See smartParse (Smart Parse) on page 755 and code-128 on page 769.
New gs1* bar code aliases.	See Bar Code type listing on page 765.

The Report Design Document

- [Overview of Genero Report Designer](#) on page 657
- [Designing a Report](#) on page 659
- [Design How-To](#) on page 676
- [Working with business graphs](#) on page 694
- [Expressions in properties](#) on page 712

Overview of Genero Report Designer

Report Designer provides you with the tools to design Genero reports.

- [Genero reports](#) on page 657
- [Launch the Report Designer](#) on page 657
- [The Report Design window](#) on page 657
- [The Report output](#) on page 658

Genero reports

Genero reports can take a variety of formats.

Genero reports can be:

- General Documents, such as invoices, corporate documents, and accounting reports; you define the contents, page size, page headers and footers, output format, and other attributes
- [Pre-printed forms](#), where you define the content for a pre-printed form.
- Labels, where you define the content and label size to be printed on pages of labels.
- [Business Graphs](#), where you specify the type of graph and the data items to graph.
- Compatibility reports, where you output a legacy Genero (4g1) report in ASCII text, using Genero Report Writer.

A report can be displayed in various [output formats](#).

Launch the Report Designer

There are several ways to launch the Report Designer.

To open the Report Designer:

- Choose **File >> New, Reports** from the Genero Studio main menu.
- Double-click an existing report design document (4rp) in the Project Manager.

The Report Design window

The Report Design window displays when you launch the Report Designer.

In the Report Design window, you create a Report Design document (.4rp). The Report Design document (4rp) defines the report page as consisting of box-shaped containers, to hold the data objects and text or image objects that make up the report.

The Report Design window consists of the work area and several views. If a view is not visible, you may need to add it using the **Window >> Views** menu.

Work Area

The work area is located in the center of Genero Studio. When you open a report, this area contains the report page.

- Report elements (containers such as [Mini Page](#) and [Layout Nodes](#), and their child elements such as [Word Box](#) and [Image Box](#)) and [data values](#) from the Data View can be dragged and dropped onto the report page and rearranged.
- [Page Headers and Footers](#) can be defined.

Structure View

The organization of the report can be seen in the [Structure View](#). [Containers](#) and their contents can be moved around in the View to insure that they print in the correct order.

[Triggers](#) in the Structure View specify what should be printed when a trigger event (change in data) occurs.

Data View

Your Report Design Document will use an `rdd` file to determine the data schema for a report. The same `rdd` file can be used for multiple report definitions.

The data schema file is specified in the [Data View](#). Once a data schema has been selected in the Data View, the available data objects are shown. The data schema is only used during report development. At runtime, the data is sent to the report by the report application.

Properties View

Each report element has properties, displayed in the [Properties View](#), with values that you can change.

In addition to literal values, [expressions](#) can be used to change the value of report elements properties. For example, the appearance can be changed conditionally, by creating an expression to turn the background color red if the value for the form element is greater than 1000.

Output View

The Output View is used by any module to display its output. The output of the various modules (messages, errors, results of commands, and so on) is displayed as tabs in the view.

Document Errors view

The Document Errors tab displays any error messages for the report design document.

The Report output

A report can be output in various formats and different page sizes, to various output devices.

Your report application can use reporting API functions to specify output details.

Default output and printer options can be set for each report design document (`4rp`).

To set default paper settings for a report, you would open the report (`4rp`) and select **File >> Report properties >> Paper Settings**. Paper settings include:

- The orientation of the page (portrait or landscape)
- The units of measure for the page (centimeter or inch)
- The page size format (standard or custom) as well as the type of paper (letter, legal, and so on)
- Margins.

To set a default output configuration for a report, you would open the report (`4rp`) and select **File >> Report properties >> Output Configuration**.

- Choose an output format, such as SVG, PDF, or Image. If you choose SVG format, it displays in the Genero Report Viewer. The Genero Report Viewer is provided as part of the Genero Desktop Client. It

provides a preview of the report, and allows the user to select a printer. To view a report output as PDF, you must have Acrobat Reader.

- Set rendering defaults.
- Set a default page range.
- If you select Image, you can then set Image Settings, to include the file type, the resolution in dpi, and the image prefix name for the created report file.

Designing a Report

The Report Designer consists of various views, menus and toolbar buttons that enable complex report designs.

- [The Report Design window](#) on page 659
- [The work area](#) on page 660
- [The Tool Box view](#) on page 661
- [Placing elements on the report page](#) on page 661
- [Changing a report element type](#) on page 664
- [Changing a property value \(The Properties view\)](#) on page 665
- [Adding report data \(Data view\)](#) on page 666
- [Organizing the report structure \(the Report Structure view\)](#) on page 668
- [Using page numbers](#) on page 672
- [Report Design Document metadata](#) on page 673
- [Configuring the output](#) on page 674

See also: [Design HowTo](#)

The Report Design window

The Report Design window displays when you launch the Report Designer.

To open the Report Design window:

- Choose **File >> New, Reports** from the Genero Studio main menu.
- Double-click an existing report design document (`4r.p`) in the Project Manager.

When creating a new report, you can begin with:

- A blank report.
- A list report template that has a basic structure already in place.
- A report from a template.

Views and windows provide the tools and work areas for the report. Use the **Window>>Views** main menu option to display and hide views:

- The [work area](#)- main window of the report
- [Structure View](#) - a tree of the report containers and their contents.
- [Properties View](#) - a list of the properties for a selected report element.
- [Data View](#) - a list of the data objects that are available for the report.
- [Toolbox](#) - a list of the containers that are available.
- The Output view - display of messages written to standard out
- The Document Errors view - a list of errors in the opened report design document or template.
- The Tasks view - a task manager showing running applications.

Note: Metadata for the report design document can be stored in the properties of the report node.

The work area

The work area in the Main Window provides a GUI interface to the report.

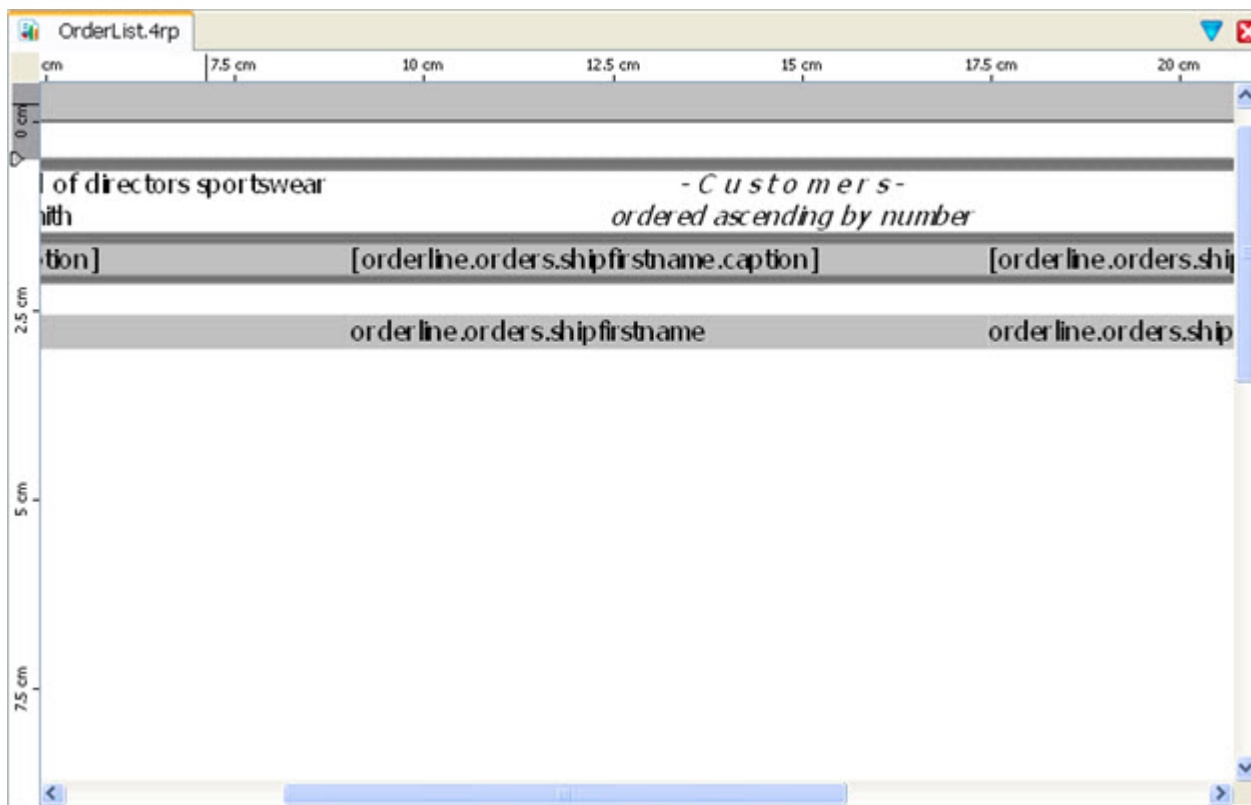


Figure 324: Report Designer work area

You design a report by initially dragging and dropping containers from the [Tool Box view](#) into the Work Area, stacking and arranging them to create the report page. Next, report elements such as Word Boxes, Decimal Format Boxes, and Images are dragged and dropped into the containers.

From the [Data View](#), you can drag and drop Data Items into a container, if you have specified the data schema.

When you select a report element in the work area, its properties are listed in the [Properties view](#). In the Properties view, you can change a property value. For example, a [WordBox](#) has a text property where you can enter text to be displayed in the report.

If you select multiple elements, all items in the current selection are affected by the current operation, such as moving, sizing, or changing the type or text.

Use the **View>>Toggle View** menu or the Toggle View icon to toggle the work area between the report design and a preview of the report. When you preview a report, [sample data](#) is displayed on the page.



Figure 325: The Toggle View icon

Zoom buttons on the Toolbar allow you to zoom in and out on the report design document.

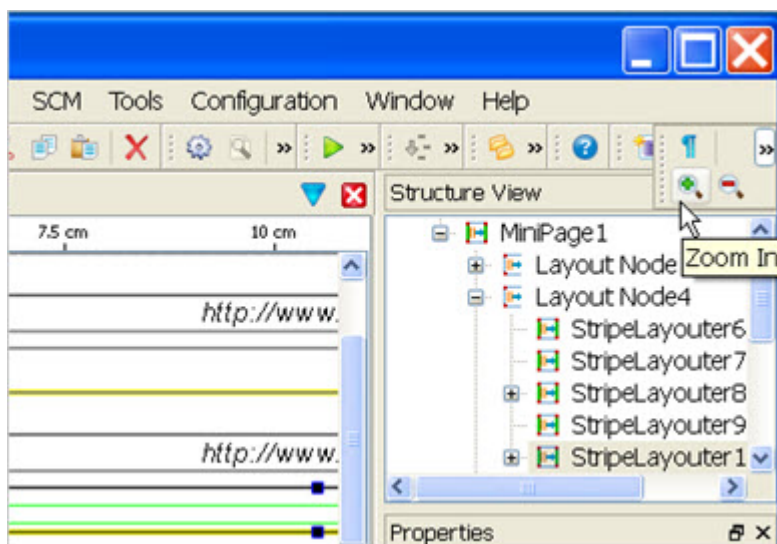


Figure 326: Zoom buttons

The Tool Box view

The Tool Box view provides report elements to place on a report design document.

The Tool Box view, typically displayed as a tab, provides the following report elements:

- Containers - for grouping other elements on the report page; see [Choosing Containers](#)
- Drawables - the report elements contained by a container; see [Choosing Other Report Elements](#)
- Business Graphs - the specific charts and items; see [Working with Business Graphs](#)
- References - to define layout-specific elements; see [Choosing Other Report Elements](#)
- Bar Codes - specific bar codes; see [Bar Code Values](#)

These elements can be [dragged and dropped](#) into a report design document.

Placing elements on the report page

When placing elements on a report page, you determine whether the positioning is specific or dynamic by setting element properties that determine how the element acts as the report changes.

Specific positioning or automatic layout

The position of a report element can be specific, to enable reports that use pre-printed forms, or dynamic, adjusting as needed based on the length of the report element.

Dynamic Layouting

if you CTRL+drag the report element onto the report page (work area), the element is positioned relative to the existing elements. Because the design changes dynamically based on the actual size of the specific report items, this is the recommended method. When you drag the element, a moving red line indicates where the element will be located when you drop it. A colored dot on the element indicates its attachment point.

If you select a container and then double-click an element in the [Tool Box view](#) or [Data View](#), the element will be automatically positioned after the last existing child object in the container.

Specific Positioning

If you drag and drop report elements on the report page (work area) using the mouse, you can position the element at a specific spot on the virtual grid of a container. This is recommended when you need to

match the report design to a pre-printed form. When you drop the element, it will snap to the closest point on the grid. A red dot on the element indicates its attachment point.

As you drag an element, a moving thin black line helps you line it up with other elements on the report if desired. The **X** and **Y** properties of the element in the Properties view indicate its location relative to its parent. These are automatically calculated when you drop an element into a container, or move it around. When you move it inside a container, the lines of the container are highlighted in yellow :

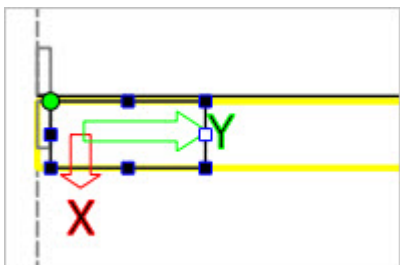


Figure 327: Highlighted container

All elements that are dragged from the Tool Box view or Data View have the **floatingBehavior** property set to "enclosed", meaning the object will be enclosed in its parent.

- X-Axis and Y-Axis arrows - These indicate the direction of the X-Axis and Y-Axis. On containers, the Y-Axis arrow indicates the filling or **layoutDirection** of the container. For example, a **Stripe** lays out its children next to each other left-to-right within the container, and the Y-Axis arrow points to the left. Other containers have the Y-Axis arrow pointing down, as they lay out their children next to each other in a top-to-bottom direction.
- Attachment point - The attachment point at an intersection of the X-Axis and Y-Axis is indicated by a green dot (for dynamic layouting) or a red dot (for specific positioning.) If you drag the edge of the element to expand its size, the attachment point remains fixed. You can move the attachment point using the right-click context menu.

Elements on a report have a contextual (right-click) menu of options that allow you to:

- Align elements within a vertical or horizontal container, and move the attachment point on the element.
- Change the width and height of elements.
- Change the focus to a different container or other element.
- Change the object type.

Drag multiple objects

You can drag and drop a multiple selection of objects from the Data View onto the report page (work area) or to the Structure View.

Use the CTRL and SHIFT keys to select the objects, then drag them to the desired location. An object will be created for each element selected, following the order in which they were selected.

If you drag to the Design work area using **specific positioning**, an additional container is created for each element object. If you have chosen to **create a form field object**, a horizontal container is used so the elements will appear in a line. When you **create a column object**, a vertical container is used so the elements will appear stacked.

If you use **dynamic layouting**, or drag to the Structure View, the behavior is the same as if each element has been dragged individually.

Choose the right container

Selecting the right containers gives you the ability to easily manage and organize your reports.

The template for new reports starts with a **Page Root** in the report structure, which is a **Mini Page** container expanded to its maximum width and length. Other containers used for the report are dropped within the

Page Root. Although you could drop all the elements directly on the Page Root, building up the report in blocks of containers allows you to group elements together, move the groups around, and align the children elements within a parent container.

- A Mini Page is used for the main container of a report page. The default name is PageRoot. The default Layout Direction when you add elements to a Mini Page is top to bottom, down the length of the page. This container propagates: when a report is printed, if a Mini Page fills, a copy is made and the leftover material flows to the copy or copies as needed.
- Use a Layout Node for page headers and footers. A Vertical Box (Layout Node) defines a rectangular area in the report in which the elements are laid out top-to-bottom by default. A Layout Node does not propagate; the contents of headers or footers can not spill over into another page. Within the header or footer Layout Node, use Stripe (Mini Page) containers for elements that should be laid out left-to-right across the page. Adding a Stripe to a Layout Node automatically extends the Layout Node across the page.
- Use a Stripe (Mini Page) container for table rows. A Stripe (Mini Page) container is a Mini Page with the Y-Size set to "max", so it stretches across the report page. Items added to a Stripe are laid out left-to-right. If the elements within a Stripe exceed the page width, the row is broken into the next line.
- Use a Mini Page for a report page with a different layout, such as a different first page.

Choose other report elements

After placing a container on a report page, elements are added for data and other report objects.

Data View

From the [Data View](#):

Icons at the top of the data view allow you to specify the type of object you wish to create when you drag and drop a data value: whether you wish to drag and drop a reference to a data value, or a data item's caption (title), or to choose to have the values and captions aligned in a table format. See [Data Values and Captions](#).

- For data values passed to the report, when you place the data item on the design page, it is automatically enclosed in a [Word Box/ Word Wrap Box](#) or [Decimal Format Box](#) container, depending on the data type.
- For captions (titles) for data items, the caption is automatically enclosed in a [Word Box](#).

Tool Box view

From the Tool Box view:

- For additional text - use a [Word Box](#) or [Word Wrap Box](#). You can enter the text when you drag the report element onto the report design page, or you can set the value of the [text](#) property in the Properties View. You can also double-click on the text in the report design page to edit it.
- For Numeric data - use a [Decimal Format Box](#), which makes it possible to parse and format numbers in any locale. The [value](#) property specifies the number. You can define the printed format, including decimal places, by setting the value of the [format](#) property in the Properties View. You can also double-click on the text in the report design page to edit it.
- For HTML pages - use an [HTML Box](#) on page 725, which displays the image of an HTML document in a report.
- For Images - use an [Image Box](#), which allows you to specify the image to be printed by setting the [location](#) property.
- For page numbers - use a [Page Number Box](#) to automatically display the correct page number for each report page.
- For tables - use a [Table](#) on page 729 to set up an object that contains columns and rows to display rows of data.
- For business graphs - choose the appropriate [Business Graphs](#) on page 731 object (chart or pivot table) for the specific type of graph. See [Working with Charts](#).

- For values like "Total from previous pages" or "Totals until this point" - use an [InfoNode](#) and [Reference Box](#). These two elements work together to enable this type of content. See [Design HowTo](#) for additional information.

Modify the sizing policy of containers

Arrow-shaped controls on the four sizing knobs located at the center of the sides of the item in focus allow you to view and modify the sizing policy of a container.

Arrows pointing inward indicate a shrinking sizing policy:

- X-Size="min" and Y-Size="min" *or*
- X-Size-Adjustment="shrinkToChildren" and Y-Size-Adjustment="shrinkToChildren"

Arrows pointing outward indicate a maximizing policy:

- X-Size="max" and Y-Size="max" *or*
- X-Size-Adjustment="expandToParent" and Y-Size-Adjustment="expandToParent"

Clicking on an arrow toggles its value.

Examples

These images illustrate some common cases.



Figure 328: Container packs the content as tight as possible



Figure 329: Container packs content vertically and expands content horizontally

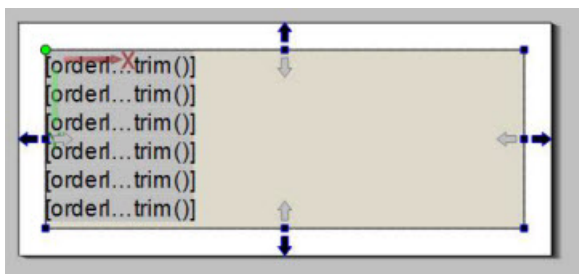


Figure 330: Container expands the content to use up the available space

A page root container typically expands in all directions to use up the available space.

Changing a report element type

To change a report element from one type to another, right-click the element in the report design document work area, and select **Convert To**; choose the new type from the list that displays. To convert the type of multiple report elements at once, Ctrl-click each element, and right-click to display the context menu.

The **name** property of the element will not change, unless the name of the old node is of the format [Type] [Number]. A node named WordBox12 would be renamed, for example.

By default the new object type will have the properties set that it has in common with the old type. For some type conversions, additional properties may be set. For example, when converting a **Decimal Format Box** to a **Word Box**, the **value** property is converted to a string value and assigned to the **text** property.

Changing a property value (The Properties view)

Select a report element in the report page (work area) or Structure View to display the property values in the Properties View. Once displayed, edit the property value.

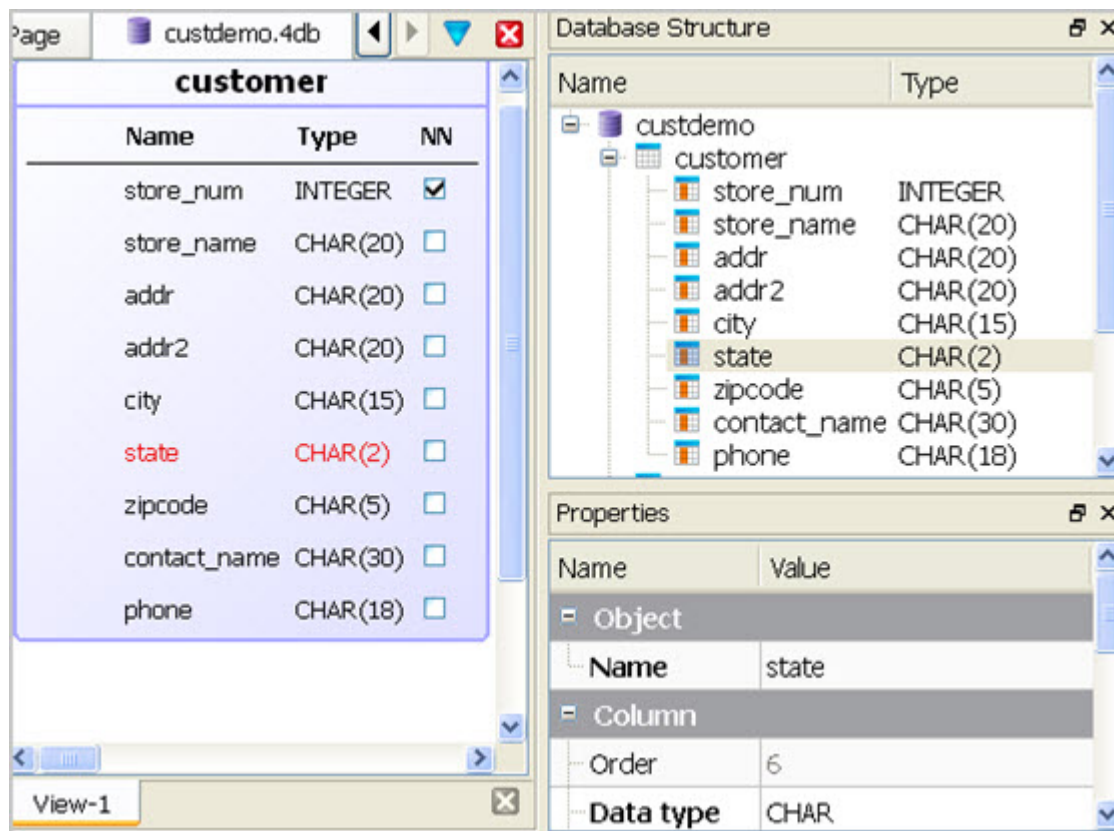


Figure 331: Properties View

The values for the properties of a report element can be changed by typing the new value in the Value column. The value may be a literal value, or it may be an **expression** written using the **RTL Expression Language**. All the properties are assigned a type, and the values entered must be valid for that type. The type of each property is listed in the **Properties** page.

Note: For WordBoxes, WordWrapBoxes, and DecimalFormatBoxes, if the text property is a literal value it may be edited directly in the report design document. Double-clicking on the element selects the text and places the input cursor in the document. The layout of the document is updated on each keystroke.

Using expressions for property values

A valid expression for a property value is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

The RTL Expression language used in Report Writer closely follows the Java™ syntax for expressions and evaluation semantics.

- Arithmetic formulas can be used.
- Conditional expressions allow you to express IF/ELSE statements.
- Genero BDL variables can be used

- Functions from the Reporting API can be used.

Press the **fx** button to open the Expression Editing Window. See [Using RTL Expressions](#) for additional information.

Adding report metadata

Five string properties for Report metadata can be specified for the document root in the Report Designer Properties View.

The metadata fields include title, author, creator, subject and keyword .

The metadata is inserted into the final document, if the output format supports metadata.

In the case of SVG, the **title** property is used as a document caption in GRV.

For reports running in compatibility mode (reports having no associated `4rp` report design document), the values can be set by calls to the corresponding API functions. See the Genero Studio >> Report Writer documentation topic "Reporting API Functions".

The "Keywords" property is currently not working for "xlsx" format.

Adding report data (Data view)

The Data View specifies the structure of the data record for the report.

The structure of the data record is defined by the input Report Schema file. The Report Schema file is extracted from the application source files:

- For applications written in Genero BDL, an `.rdd` file is extracted.
- For arbitrary XML data sources, you can generate an `.xsd` file to describe the data schema. See [Support for arbitrary XML data sources](#) on page 667.

Within the Data View:

- The **Arrows** icon allows you to sort the data items alphabetically.
- Values in the **Sample Data** column display when you preview a report. Double-click a value to edit or replace it.
- The **Filter Fields By Name** field, located at the bottom of the Data View, allows you to specify filtering criteria for the Data View, where only fields containing the name entered in the box are displayed in the data items.

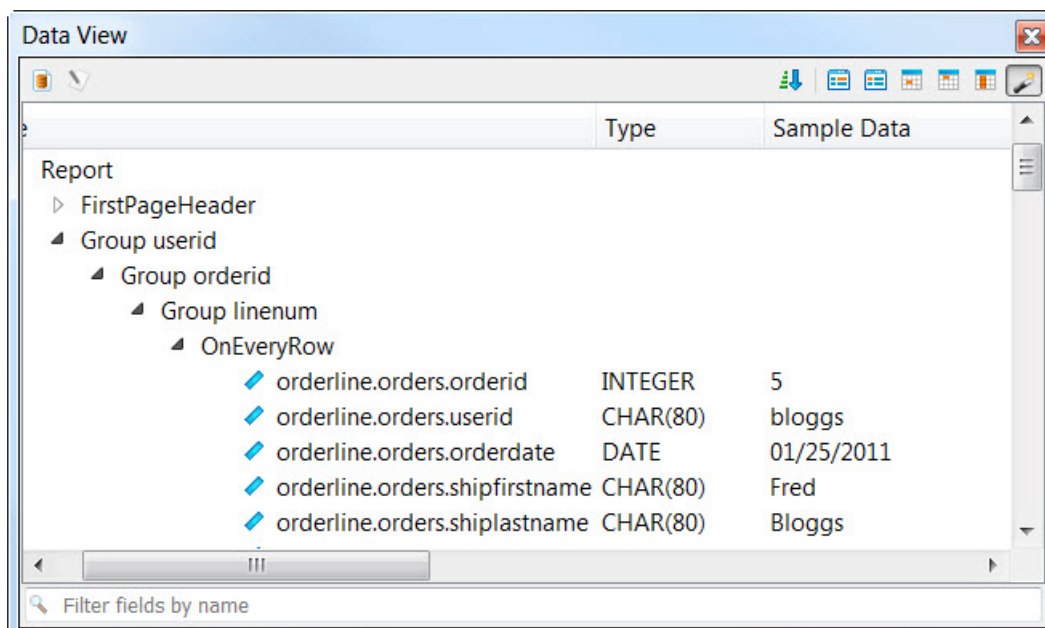


Figure 332: Data View

Click the **Open Data Report File** icon at the top of the Data View to specify the Report Schema file to populate the Data View.

Adding data values and captions

Before you place a data object onto the report design window, click one of the icons on the integrated toolbar to specify whether you wish to drop the item or its title, and whether the object is part of a table:

Dropping the object as a simple report field:

- Form field value object
- Form field title object - Places the caption for the selected object. Use for field labels.

Dropping the object aligned as part of a table (the space allocated for the column will be the larger of the space required for the data or the title, helping to align the title and data in the columns of a table):

- Table column value object
- Table column title object - Places the caption for the selected object. Use for table column headers.
- Table column value object for a column without a title - if you are not going to have a column header, the space allocated for the column is set to the maximum required by the value only.

Allow the Report Designer to determine the type of dropped object.

- Create element based on the document context

The object type created for a field is determined by the location in the document. Consider dragging a numeric field to two different locations in a document. In the first instance, the object is dropped into the OnEveryRow stripe and it becomes a Decimal Format Box. In the second instance, the object is dropped onto a Map Chart and it becomes a chart Item element.

Table 171: Rules governing element creation based on context

Element	Condition
ITEM ("key" is set if field isn't numeric, "value" otherwise)	Parent element is a MAPCHART
CATEGORYITEM ("key" is set if field isn't numeric, "value" otherwise)	Parent element is a CATEGORYCHART
Same object as option "Create a table column title object"	Parent element is of class <code>grwTableHeader</code>
Same object as option "Create a table column value object"	Parent element is of class <code>grwTableRow</code>
Same object as option "Create a table column value object for a column without title"	Parent element is of class <code>grwHeadlessTableRow</code>
Same object as option "Create a form fields value object"	If none of the above are applicable.

The data objects are automatically contained in a [Word Box](#) if the data type is defined as less than 30 CHAR, and in [Wordwrap Boxes](#) if the data type is defined as larger than 30 CHAR. If the data type is Numeric, the data object is contained in a [Decimal Format Box](#).

The [text](#) property of the Word Box or Word Wrap Box, or the [value](#) property of the Decimal Format Box, specify what will print in the report output. The value property of the Decimal Format Box can be calculated using an Expression. See [Using RTL Expressions](#).

Support for arbitrary XML data sources

Genero Report Writer can produce reports from arbitrary XML input sources.

The xml source is described by an XML Schema. The [Open schema file](#) dialog proposes `rdd` and `xsd` file formats.

If an `xsd` file is selected, the designer interprets the file as follows:

- Any XML Attribute is considered a variable.
- Any simple type element with `minOccurs=1` and `maxOccurs=1` is considered a variable.
- Any complex type elements with `minOccurs=0` or `maxOccurs>1` produce triggers.

For an example, see the `Table.4rp` report design document in the **Reports** sample project, located under the **OrderReportXML** application node. Open this report to see the `OrderData.xsd` used as data schema for the report (as shown in the Data View tab).

Encoding null values in the data

The attribute `xsi:nil` (with `xsi` representing the namespace `http://www.w3.org/2001/XMLSchema-instance`) can be used on an empty element to denote a null value. The RTL function `isNull()` for input variables will return true for such a variable. For example, consider the following document fragment:

```
<input xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
<productReference xsi:nil="true"></productReference>
```

The RTL expression `productReference.isNull()` will yield true for this instance of the variable.

Limitations

Recursive references are not supported. Elements involved in a recursive reference will be ignored by the designer.

Optional variables are not supported. A variable is optional if one of these conditions is true:

- It is an XML simple type element with `minOccurs="0"`.
- It is an XML Attribute without `use="required"` in the schema definition.

Simple type element with `minOccurs="0"` are discarded by the designer. No variable is created.

The designer creates variables for optional attributes and issues a warning. An error occurs if the variable is not present at runtime.

The Genero application

When you declare the data of a report design document (`.4rp`) to use an arbitrary XML data sources, you must use the `fgl_report_runFromXML` API function in your report. See [fgl_report_runFromXML](#) on page 626.

Organizing the report structure (the Report Structure view)

How do you make sure that the various elements print in the correct place in the report? Use containers - by combining horizontal boxes, stripes, word boxes, etc., in the correct order in the parent container, you control the order in which the elements in that parent container print out. Then you can move the containers around in the Structure View to make sure they print out in the correct spot.

The tree structure

The Structure View is a tree containing object and trigger nodes.

object nodes

In the Structure View, the object nodes are the containers and objects that are to be printed on the report output.

trigger nodes

In the Structure View, the trigger nodes are the triggers (event handlers) that specify when the object nodes are to be printed out.

Change the hierarchy of the objects and triggers in the Structure View by dragging and dropping them within the tree:

- Drag a container or trigger and drop it on a different node; it will become a child of that node.
- Press the ALT key and drag a container or trigger, dropping it on a different node; it will become the parent of that node.

Triggers

The data for your report is passed from your report program to the report one row at a time, sorted by the criteria specified in your report.

- For a Genero BDL program, the SQL statement defines the sort criteria.

Each row streamed to the report contains all the data or text specified by the data schema (the data report file). You may want to print some of the data from each row received; this is generally the report body. Other data and text should only print when a change in a specific data item takes place; for example, you may want to print a total each time the `customer_id` value changes. The *triggers* in the structure of the report specify what should be printed when a change in data occurs. Arrange all the report elements for a single trigger in a parent container.

Triggers and Genero BDL report applications

Trigger nodes are specified by the `ORDER EXTERNAL` statement in the `REPORT` program block in your Genero BDL code, and indicate the data values by which the data is grouped. The final trigger node is `ON EVERY ROW`, specifying what is to be printed for each row of data passed to the report. The trigger nodes display in the Report Structure as red bullets:



Figure 333: The ORDER BY clause and report triggers

When the Data Schema changes

When the data schema associated with a report is modified, the Genero Report Designer regenerates the triggers to match the new schema. The Genero Report Designer will attempt to update the Report Structure using the minimum number of modifications required to perform the update. Cases of adding new triggers and cases of removing triggers that do not contain document fragments are handled automatically.

When it is necessary to move or remove a trigger that contains a document fragment, a warning displays in the Document Errors view stating that the fragment may require manual correction. These warnings are stored persistently in the report definition (`4rp`) file. The warning can only be removed by using the context menu **Clear trigger update message (issue is fixed)**.

The warning message `Data schema changed. The document has been updated to match the changes` is only displayed when a new trigger is inserted, a trigger is removed or a trigger is moved.

Place a trigger within the report structure

Report triggers appear in the report structure after you select the data schema for your report design. You can organize your report structure using drag-and-drop within the Report Structure view. Alternatively, you can use the **Repeat selected items on** contextual menu to easily make a trigger the parent of a document node.

1. Identify the node in the report structure that you wish to be the child of a specific trigger node.
2. Right-click the node and select **Repeat selected items on**.
The list of triggers appear in a sub-menu.
3. Select the trigger.

The trigger becomes the parent of the currently selected document fragment. The structure of the document tree is preserved.

Note: A similar functionality is available by holding the ALT key while dropping a trigger on top of a document node.

Page headers and footers

Arrange all the elements for a Page Header or Page Footer in a parent container that does not propagate.

A Vertical Box (Layout Node) is typically used. A layout node container has a [section](#) property. In this property, you specify the header or footer you are defining: first page header, any page header, last page header, first page footer, and so on.

Example report structure

Use the Structure View to examine the structure of your report.

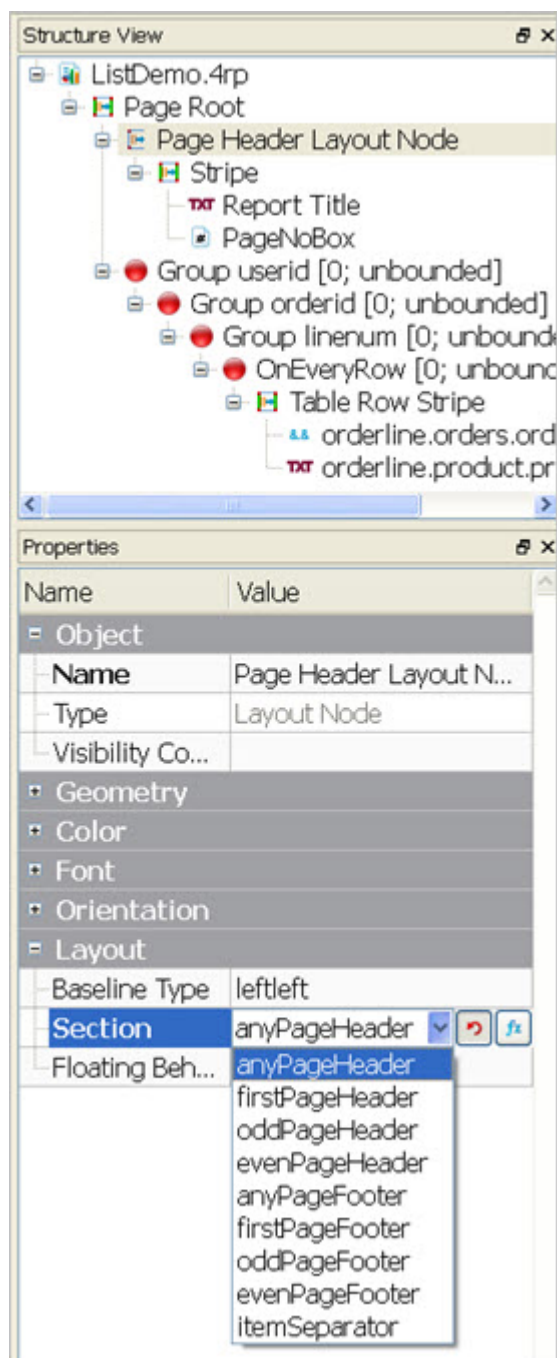


Figure 334: Structure View and Properties of a sample report

The Page (Page Root container)

- Page Header: In the example shown, the Page Header is a Layout Node container that specifies what is to be printed as a Page Header. See [Page Headers and Footers](#). The example Page Header contains:
 - a Stripe container
 - a Word Box named Report Title
 - a Page Number box

- The example data-related [triggers](#) Group userid, Group orderid, and Group linenum do not cause anything to print as they have no child containers.
- The example OnEveryRow trigger has a Stripe containing a Decimal Text Box and a Word Box. These are the data items that will be printed for every data row that was passed to the report.

The Printed page

Placement of the header and footer containers take the page size into account.

When a report page is printed, any Page Header (if defined) will print at the top of the page, followed by the content of the containers associated with trigger nodes, followed by the Page Footer (if defined) at the bottom of page, in accordance with the [Page Size](#) set for the report. This is the basic design of the report page, which will be repeated when the report is run for as many pages as are required, based on the data passed to the Report Writer.

Using page numbers

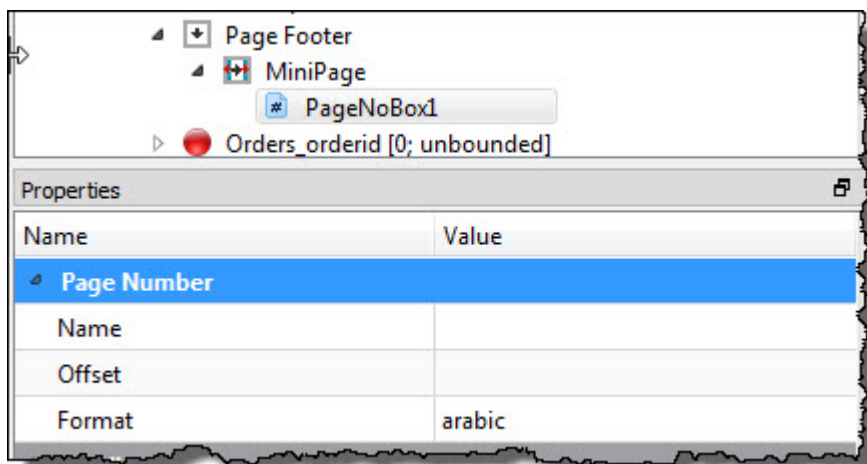
Use the Page Number drawable to print page numbers on a physical page.

The [Page Number](#) drawable (PageNoBox) is a layout container that prints the page number of the physical page. It is frequently part of a Page Footer. The drawable prints only the page number (a numeric) by default, but you can use the **Text Expression** property (textExpression) to create a page number string, such as the standard "Page N of M". See [Using a page number string](#).

Use the Page Number container properties [Offset](#) (pageNoOffset), [Format](#) (pageNoFormat), and [Text Alignment](#) (textAlignment) to format the page.

Use the **Name** (pageName) property if you want to reset the page number each time a specific report trigger fires. Select a MiniPage under the report trigger to use as a basis for the page number count. You can add a new MiniPage for this purpose if necessary.

Figure 335: Page Number Box properties



Genero Report Writer automatically calculates a Page Number container size based on a four digit page number or you can set the size with properties such as **X-Size** or **Text**. Set the length explicitly using the **X-Size** property (e.g. "3cm"), or use the **Text** property to hint a smaller size. For example, set the **Text** property to "000" to specify a maximum length of 3 digits. If multiple sizing properties are configured, Genero Report Writer uses the setting with the highest priority as follows (listed in highest to lowest priority):

1. X-Size
2. Text
3. Text Expression

Using a page number string

For Page Number containers, use the Text Expression property to create a page number string, such as the standard "Page N of M".

If you set the **Text Expression** (textExpression) property, it's important to verify that the Page Number container length is long enough to print the full page number text. The size of the string depends in part on the number of page breaks in the report. For example, the string "Page 2 of 5" has a smaller length than "Page 2 of 76".

Because of the unpredictable variation in size, Genero Report Writer defaults to a container size based on a four digit current page and total number of pages (i.e. "Page 9999 of 9999"). To override the default behavior you can set the **X-Size** to an explicit size or hint a smaller size with the **Text** property (e.g. "Page 99 of 99").

The values of the **Name** (pageName), **Offset** (pageNoOffset), and **Format** (pageNoFormat) properties are ignored when the **Text Expression** property is set.

These functions can be used to format and access specific page numbers and totals.

- Class String: `format(Numeric number, Enum format)` - formats the number as specified. The value for the format parameter can be ARABIC, LOWERROMAN or UPPERROMAN.
- Class Numeric: `getPhysicalPageNumber()` - gets the current page number of the physical page.
- Class Numeric: `getTotalNumberOfPhysicalPages()` - gets the total number of physical pages.
- Class Numeric: `getPageNumber(String pageName)`- gets the page number of the specified page
- Class Numeric: `getTotalNumberOfPages(String pageName)` - gets the total number of pages for the specified page.

Note: If you use functions `getTotalNumberOfPhysicalPages()` or `getTotalNumberOfPages()`, report pages waiting to be updated with the actual page count will be held back if printing is initiated from the viewer.

Examples

This expression computes the string "Page N of M" for the physical pages. The equivalent of the "Offset" property can be achieved by doing arithmetic with the results from the page number functions. In this case, the numbering will start at page 11 since the example formula adds 10 to the value returned from the function `getPhysicalPageNumber`.

```
"Page "+format(getPhysicalPageNumber(),ARABIC)+" of
"+format(getTotalNumberOfPhysicalPages(),ARABIC)
```

This expression computes the string "Page N of M" for logical pages, providing page numbers for each order within a batch of several orders.

```
"Page "+format(getPageNumber("pageRoot"),ARABIC)+" of
"+format(getTotalNumberOfPages("pageRoot"),ARABIC)
```

Report Design Document metadata

The Title, Author, Creator, Subject, and Keywords properties of the document root allow you to add metadata for a report.

Select the document root in the Structure View. Enter your values for the properties.

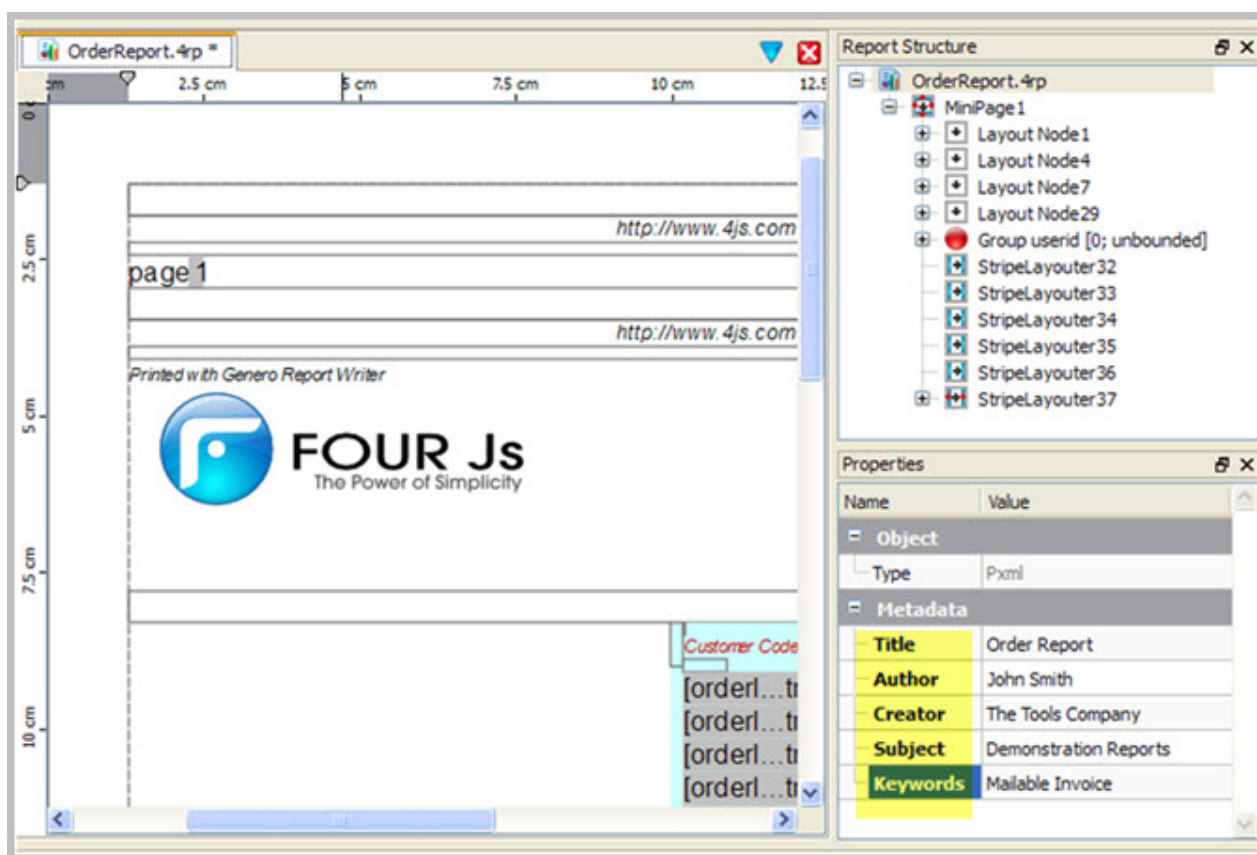


Figure 336: Report Design metadata

The metadata is inserted into the final document (PDF, SVG) if the format supports metadata. In the case of SVG, the **title** property is used as a document caption in the Genero Report Viewer.

If your report will run in compatibility mode (having no 4rp design document), the values can be set by calls to the corresponding Report API functions: `fgl_report_setTitle()`, `fgl_report_setAuthor()`, `fgl_report_setSubject()` and `fgl_report_setKeywords()`. See the Genero Studio >> Report Writer documentation topic "Report API Functions".

Configuring the output

The same report can be output in different formats and different page sizes, and to different output devices.

In **Preferences** you can change the default paper settings for all reports: Tools>>Preferences, Report Writer, Paper Settings. See the Genero Studio >> Report Writer documentation topic "Report Writer Preferences".

The **File>>Report Properties** main menu option allows you to change the default report options for the currently open report design document.

- **Paper Settings** - select the page size and other paper settings for a report

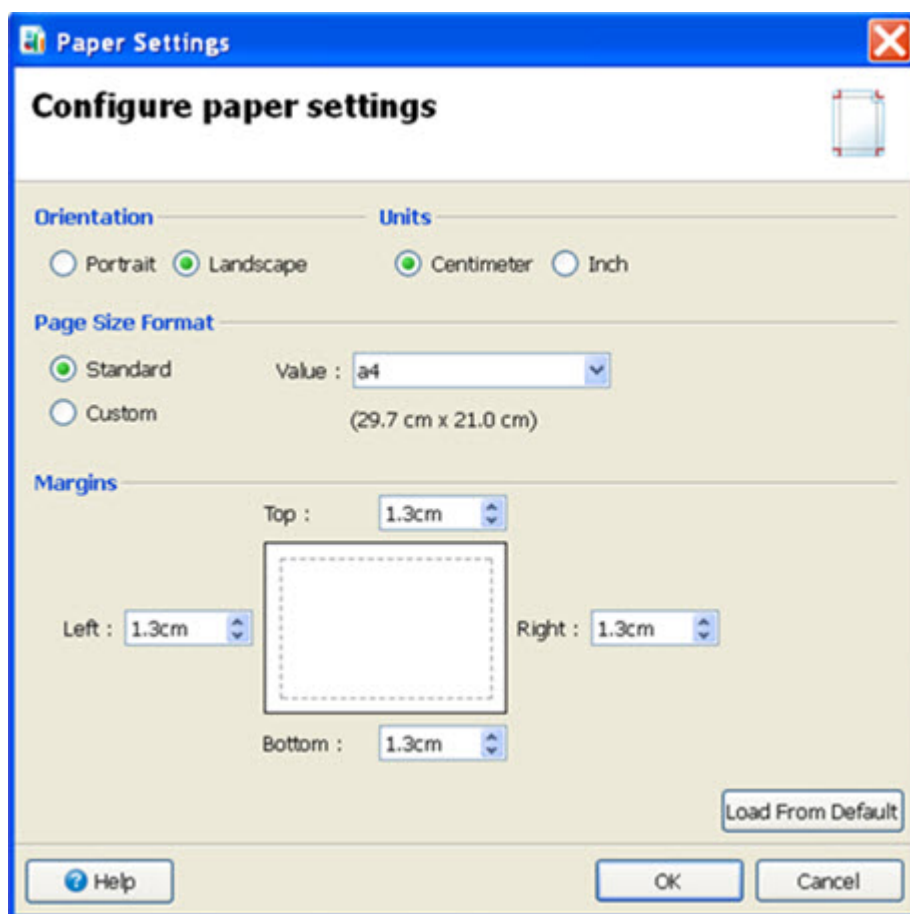


Figure 337: Paper Settings dialog

- **Orientation** - portrait or landscape
- **Units** - centimeter or inch
- **Page Size Format** - select from a list of common formats, or enter a custom height and width
- **Margins** - set left, right, top, and bottom margins

The **Load from Default** button restores the default values for paper settings as set in the Genero Report Writer Preferences. See the Genero Studio >> Report Writer documentation topic "Report Writer Preferences".

- **Output Configuration** - select the output format and other options

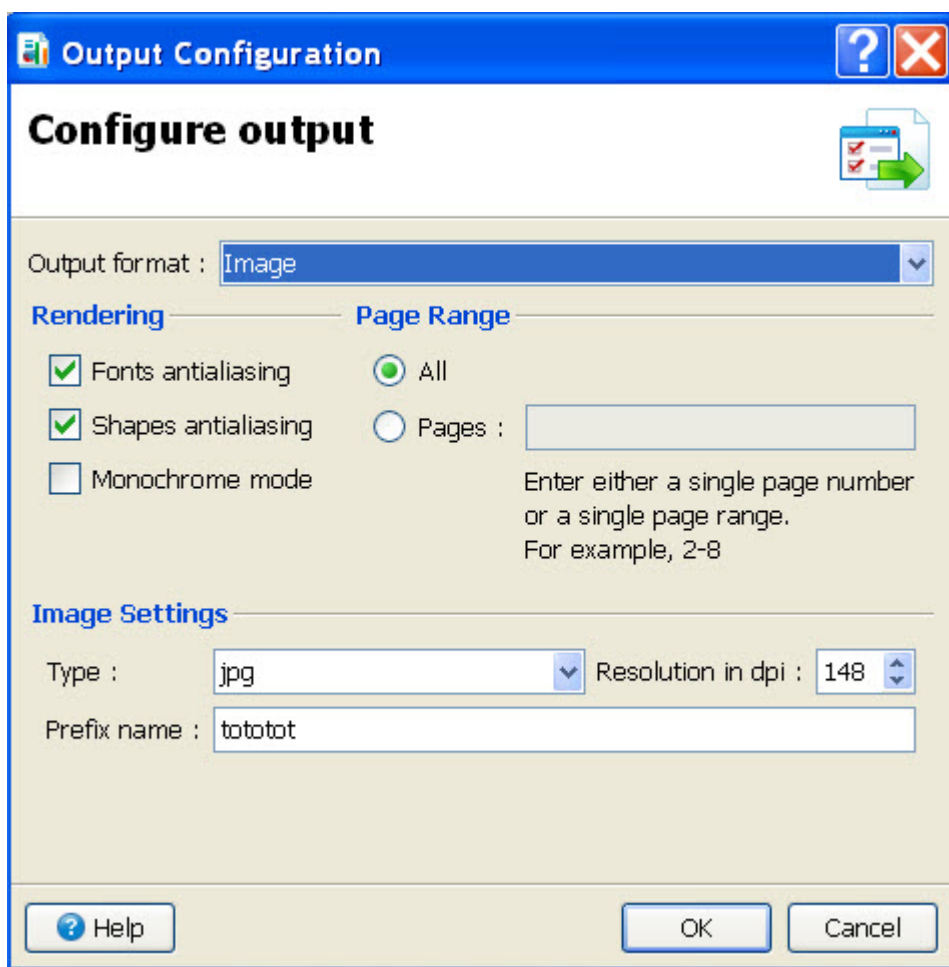


Figure 338: Output Configuration dialog

Options for **Output format**:

- **SVG** (scalable vector graphics - can be displayed using the Report Viewer feature of GDC)
 - **Rendering** - Select options to minimize the aliasing distortion
- **PDF** (Acrobat PDF format, can be displayed using PDF viewer)
 - **Rendering** - Select options to minimize the aliasing distortion or set monochrome mode
 - **Page Range** - output all pages or enter a range
- **Image** (creates an image, such as .jpg. You can select the image type.)
 - **Rendering** - Select options to minimize the aliasing distortion or set monochrome mode
 - **Page Range** - output all pages or enter a range
 - **Image settings** - Select image **Type**, **Resolution**, **prefix for the image** filename

Note: The Rendering options for font antialiasing (SVG or PDF documents) will only take effect if the [fidelity](#) property of report text elements is set to "True".

Functions from the Reporting API can be used in your report program to override the default options at runtime. For example, the function `fgl_report_selectdevice` provides additional output formats. See the Genero Studio >> Report Writer documentation topic "Report API Functions".

Design How-To

These procedures help you complete specific report design tasks.

- [General Design](#)

- [Align Numbers \(format\)](#)
- [Center Elements](#)
- [Set the paper settings of a report on page 678](#)
- [Force a Page Break](#)
- [Print Headers and Footers](#)
- [Print Group Totals and Report Totals](#)
- [Print Totals at the beginning of the report](#)
- [Have Different first and last Pages](#)
- [Print an Invoice Page Number instead of the physical page number](#)
- [Print a layout-dependent Reference \(InfoNodes\)](#)
- [Design Documents for preprinted forms on page 684](#)
- [Design labels on page 684](#)
- [Design address labels on page 686](#)
- [Modify an object's Borders, Margins, and Padding](#)
 - [Size Expressions for Bordered Boxes](#)
- [Using Hyperlinks in a Report](#)
- [Some Tips for Legacy Report Designers](#)

General design

These topics describe some common features of reports and how to do them.

Note: The basics of report design is discussed in [Designing a Report](#).

Align and format numbers

Use proper containers and properties to align and format numbers.

Use the ToolBox object [Decimal Format Box](#) as the report element for numbers. It supports different locales and kinds of numbers, including integers, fixed-point numbers, and currency amounts (\$123).

The [text](#) property specifies the value for the number to be printed. The value of this property may also be edited directly in the report design document by double-clicking the Decimal Format Box. The input cursor will be placed in the document, and the layout of the document is updated on each keystroke.

The [format](#) property specifies how the number will print out. The default value for this property is "---,---,---&.&&". You can change this string, using the specified symbols. The - (minus) symbol represent digits (if the number is negative, it will print with a leading -), and the period represents the decimal point. The & symbol fills with zeros any position that would otherwise be blank. If the actual number displayed requires fewer characters than the format string specifies, numbers are right-aligned and padded on the left with blanks.

Table 172: Formatting examples

Value of text property	Appearance in report
123456.1	123,456.10
15.24	15.24
-1600	-1,600.00

Center elements

Use properties to center elements.

To center an element in its parent container you can set its properties as described in [Table 173: Centering elements](#) on page 678

Table 173: Centering elements

Property	Value
x	max/2
y	max/2
anchorx	0.5
anchory	0.5

An **x** and **y** value of **max/2** sets the x and y coordinates of the element to the maximum of its parent container divided by 2. An **anchorx** and **anchory** value of **0.5** sets the attachment point to the center of the element.

Set the paper settings of a report

Paper settings set the paper orientation, the page size format, and the report margins for a report.

Each report defines these paper settings:

Orientation

Select *Portrait* or *Landscape* as the layout for the report.

Units

Specify whether to use *centimeters* or *inches* when defining the page size and margins for a report.

Page Size Format

Set the page size using either *Standard* sizes or *Custom* sizes. With standard, select the desired standard size from the list provided in the combobox. For custom, specify the height and width of the report.

Margins

Each page has four margins: top, bottom, left, right.

Paper Settings order of precedence

There are three areas which determine the paper settings for a report, listed here in the order of precedence:

1. In the report application using reporting APIs.
2. In the report design document (.4rp).
3. GRW default values.

Set paper settings using the reporting APIs

A report application can use the reporting APIs to override the paper settings for an individual report.

- For Genero BDL applications, see [Change paper settings and output format](#) on page 589.

Set paper settings for a report design document (.4rp)

The Genero Report Writer preferences specify the initial paper settings for a new report.

To change the paper settings:

1. Open the report in Genero Report Designer.
2. Select **File >> Report properties, Paper Settings...** The **Paper Settings** dialog opens.
3. Set the desired orientation, units of measure, page size format, and margin settings for the report.

Tip: To set the paper settings to the current values set in the **Paper Settings Configuration** of the local Genero Report Writer, click **Load from Default**.

4. Click **OK** to save your settings and close the **Paper Settings** dialog.

Set paper setting preferences for GRW

The **Paper Settings Configuration** page sets the default paper settings for all new reports.

1. Select **Tools >> Preferences, Report Writer, Paper Settings**.
2. Set the desired orientation, units of measure, page size format, and margin settings.

Tip: To set the paper settings to the installation default values, click **Load from Default**.

3. Click **Apply** to save your preferences.
4. Click **OK** to close the **Preferences** dialog.

Once created, the paper settings for a report are stored as part of the report design document (.4rp). Changes to the **Paper Settings Configuration** have no effect on existing reports.

Force a page break

To force a page break, you set the X-size or Y-size property to "rest".

To force a page break, you can insert a container in the page, using the *rest* variable in the X-size or Y-size property of the container; this makes the container consume the remainder of its parent container. The property to be set is determined by the default layout direction of the container.

To force a page break after a MiniPage or Stripe container, set the height (**X-size**) to "rest".

To force a page break after a Layout Node, set the **Y-size** to "rest".

Table 174: Forcing a Page Break

Property	Value
X-size	rest
Y-size	rest

Switch child and parent nodes

In the Report Structure, you have two nodes in a parent-child hierarchy. You want to make the child the parent, and the parent the child.

While holding the ALT key, drag the child and drop on to the parent.

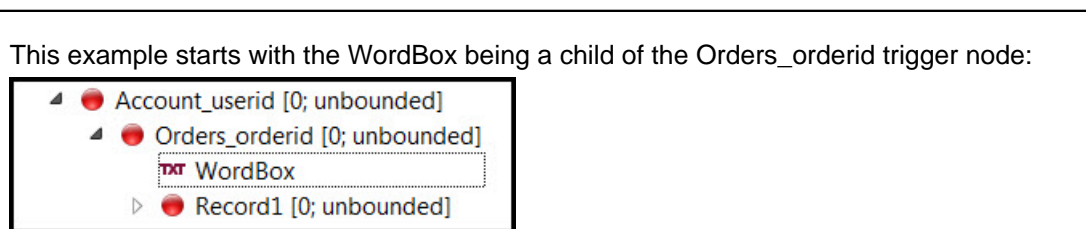


Figure 339: WordBox is child of Orders_orderid trigger

While holding the ALT key, click on WordBox, then drag and drop it on top of Orders_orderid.

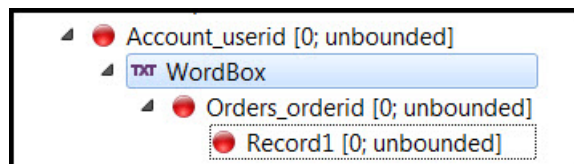


Figure 340: Orders_orderid trigger is child of WordBox

The two nodes have effectively switched places.

Print headers and footers

A MiniPage or PageRoot in your Report Design document defines how the content is to be laid out on the report page. Headers and Footers can be defined for the report pages, which can depend on the relative position of the page in the overall report.

You can use any Simple Container or Drawable as a container for a header or footer. Generally, a Vertical Box (LayoutNode) is used.

Add the header or footer container to a [MiniPage](#) or [PageRoot](#). To identify the containers as headers or footers, and to specify where on the report page a container should print, set the container's [section](#) property to one of these ports:

firstPageHeader	Page Header, to print on the first page only; if this section is defined, subsequent Page Headers begin printing on the second page.
oddPageHeader	Page Header, to print on odd pages; has precedence over anyPageHeader.
evenPageHeader	Page Header, to print on even pages; has precedence over anyPageHeader.
anyPageHeader	Page Header is for every page, unless separate Odd and Even Page Headers are defined.
firstPageFooter	Page Footer, to print on the first page only; if this section is defined, subsequent page footers begin printing on the second page.
oddPageFooter	Page Footer, to print on odd pages; has precedence over anyPageFooter.
evenPageFooter	Page Footer, to print on even pages; has precedence over anyPageFooter.
anyPageFooter	Page Footer, to print for every page, unless separate Odd and Even Page Footers are defined.

For example, a Vertical Box with the [section](#) property set to the firstPageHeader section will print as the header on the first page of the report. In the parent Container, you cannot have multiple header or footer containers set to the same section.

If you use a Vertical Box for the header or footer container, set the [Layout Node](#)'s [X-size](#) property to *max*, and its [Y-Size](#) property to *min*. Within the container you can build up the header or footer using various containers and report elements. The [Stripe](#) container is useful for report elements that are to be laid out left to right across the page. Use [Stripes](#), [WordBoxes](#), etc. as needed, to arrange the contents of the header or footer within the container in the order in which it should be printed.

The properties [HidePageHeaderOnLastPage](#) and [HidePageFooterOnLastPage](#) provide flexibility in the printout.

Important: It is an error if any element having the section set is preceded in the same sibling list by one that doesn't. In other words, any sections for the MinPage need to be specified first. Verify in the [Structure view](#) that the report structure is correct.

Print group totals and report totals

In the Genero BDL report program

In your Genero BDL report application, define variables for the totals, calculate the values, and output them to the report engine.

In the REPORT program block, define variables:

```
DEFINE
  data store_order_data,
  --Add variables for totals
  store_total, order_total, item_total, report_total DECIMAL(10,2)
```

Identify the data columns by which the data is grouped in the ORDER EXTERNAL command:

```
ORDER EXTERNAL BY
  data.orders.store_num,
  data.orders.order_num
```

Calculate the group and report totals, and output them in the FORMAT section.

```
FORMAT

  --Add FIRST PAGE HEADER, BEFORE GROUP OFs
  FIRST PAGE HEADER
    LET report_total = 0

  BEFORE GROUP OF data.orders.store_num
    LET store_total = 0

  BEFORE GROUP OF data.orders.order_num
    LET order_total = 0

  ON EVERY ROW
    --Add statements to calculate totals
    LET item_total = data.items.price*data.items.quantity
    LET order_total = order_total + item_total
    LET store_total = store_total + item_total
    LET report_total = report_total + item_total
    PRINT data.*, order_total, store_total, report_total
```

In the report design document (.4rp)

- In the [Report Design window](#), define [Stripes](#) in the design document, one for each total. Use [Decimal Format Boxes](#) to hold the values.
- In the [Structure View](#), drag each Stripe and drop it onto the [trigger node](#) for the corresponding group trigger. This will position the Stripe as a child of the trigger node. It is important that the total stripes are the last child element of a report trigger node.

Consider this example:

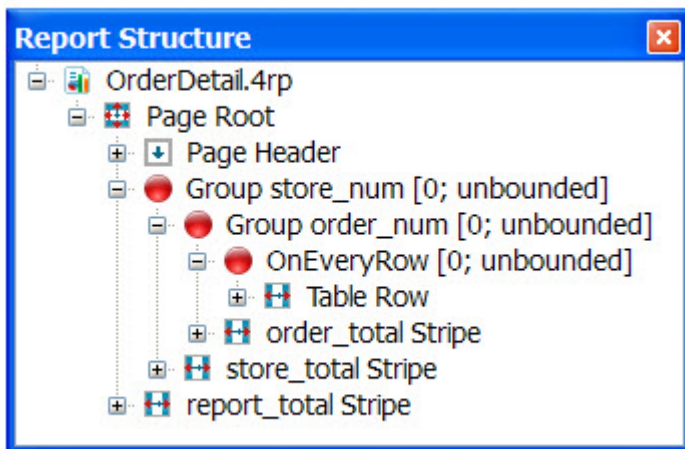


Figure 341: Example Report Structure

In this example, the **order_total** Stripe is the last child of the Group **order_num**, the **store_total** Stripe is the last child of the Group **store_num**. The **report_total** Stripe was dropped onto the **Page Root**, the main page for the report. This positions the Stripe as a child of the Page Root, and will place it at the very bottom of the child list for that trigger.

On each change of data, the specified data for every corresponding row will print out, and the appropriate Stripe will print out after each change of Group. The **report_total** Stripe will be the last thing to print out on the report. The values of any data objects in the Stripe will be taken from the immediately preceding row of data (the last report line of the container for onEveryRow trigger.)

Print totals at the beginning of a report

You can print aggregate values before the first detail row is printed.

For example, totals can be printed at the beginning for a report.

When this feature is used, the output has to be delayed until the input has been processed to the point where the variable value is shipped. In the case of a grand total, which is shipped at the end of the report, the entire input must be consumed before the document fragment containing the total would be output. If the total number of records is small, the delay will hardly be noticeable; for example, when you print the order total before printing up to a few hundred rows relating to the order.

Print a Layout-dependent reference (InfoNodes)

InfoNodes allow you to print a value on the report that depends on the paged stream resulting from the report layout.

For example, a value for "total from previous page" can vary depending on how the page options for a report are set. In order to have a report layout that will work with various page sizes, you can use an [InfoNode](#) and a [Reference Box](#).

This example illustrates how to print the total price (`overalltotal`) from a previous page.

In the Genero BDL report program

The `REPORT` block of the Genero BDL file must calculate the desired value and output it to the report. The following example is from the `OrderReport.4gl` file in the demo sample programs:

```
ON EVERY ROW
  LET lineitemprice = orderline.lineitem.unitprice *
  orderline.lineitem.quantity
  LET overalltotal = overalltotal + lineitemprice
  LET ordertotal = ordertotal + lineitemprice
  PRINT orderline.*, lineitemprice, overalltotal, ordertotal
```

The variable `overalltotal` contains the running total price of the lineitems on the report.

In the report design document (.4rp)

You will use these objects from the Toolbox in your report design:

- **InfoNode** - place this object in the container for the **ON EVERY ROW** trigger of your Structure view. This will create an invisible column in your report line containing the value of the InfoNode.

The **Value** property of the InfoNode must be a String. You can use the `fglValue` member of the **FGLNumericVariable class** to convert **overalltotal**:

```
overalltotal.fglValue
```

This will format the value of **overalltotal** as a String based on the default format set in the Genero DVM. Or, you can use the `format` method of the **Numeric class** to convert to a string and also specify the format, as in this example:

```
overalltotal.format("- ,--- ,--- ,--&.&&")
```

- **Reference Box** - place this object in the Page Header at the top of the report structure.
 - For the **InfoNode name** property, enter the name of the InfoNode that you created.
 - For the **text** property, enter a string that will only be used to determine the maximum length of the value in the InfoNode, since the value will not be known at the time the ReferenceBox is positioned. Examples: Enter "000,000.00" as the maximum length for a value that is from a numeric data type, or "MMMM" as the maximum length for a value that is from a CHAR(4) data type.
- **WordBox** - optionally use this object to add some text next to the Reference Box.

A Reference Box points to the immediately previous occurrence of the InfoNode value in the paged stream. Because you placed the Reference Box in a Page Header, it will point to the last occurrence of the **overalltotal** value on the previous page.

Specify different first and last pages

Use MiniPage containers to specify different first and last pages.

How do you have a different first or last page in a report?

- Add a separate **MiniPage** container for each page variation (first-page, main-report-page, last-page, for example), as children of the Page Root container.
- Add the report elements that are specific to each container.

In the **Structure view** the MiniPage containers should be listed in the order in which you want them to appear in the report. For example:

- Page Root
 - first-page - this is a "before" page, to print before the main content
 - main-report-page - this would contain all the triggers and containers that make up the body of the report
 - last-page - this is an "after" page, to print after the main content

Print an invoice page number instead of the physical page number

Print an invoice page number instead of the physical page number.

- Use the **name** property of the MiniPage container of the invoice to assign a name to the page.
- Add a **Page Number** report object to the page header of this MiniPage container.
- In the **Page Number** section of the properties for the Page Number object, set the **name** property to the name of the MiniPage Container.

The page number of the MiniPage will be printed on the report.

Design Documents for preprinted forms

A simple drag and drop of the report elements will invoke the Positioning method, which allows you to place an element in the desired location on the form.

To make this easier, add an Image box to the report, containing an image of the form to serve as a background.

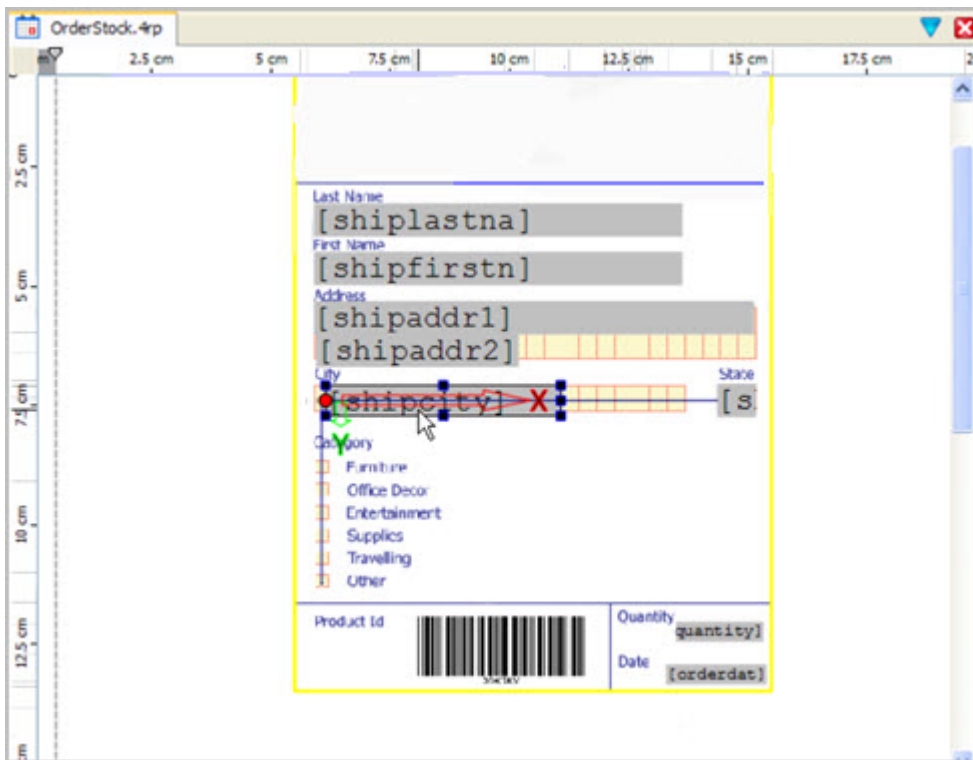


Figure 342: Report with Image

The report design window contains a global grid; as you drag a report element onto the design window, black grid lines help you align it with the other elements on the form. The red attachment point indicates the mapping of the element to the global grid.

Once you have dropped an report element, you can refine its location by dragging, or you can select an element and use the keyboard arrow keys:

- Pressing an arrow key moves the element incrementally in the corresponding direction
- Ctrl-arrow key moves the element along the global grid.

Right-click an element to display a menu of additional options.

Design labels

For a report application that prints out labels, the report design document (.4rp) is the size of a single label.

A report program programmed to output labels expects the report design document to represent a single label.

- To code a Genero report application that creates labels in a report format, see [Create labels: the report program \(Genero BDL\)](#) on page 580.

1. Create a new report.
Select **File >> New, Reports, Empty Report (.4rp)**.
2. In the **Data View**, specify a Data Schema.
See [Adding report data \(Data view\)](#) on page 666.

3. Set the page size to the size of a single label.
 - a) Select **File >> Report properties >> Paper Settings...**
 - b) Set the **Page Size Format** to **Custom**.
 - c) Set the paper settings to the size of one label. Adjust margins as needed.
4. In the custom page you've created, design the label as you would design any report, to include adding fields from the Data view.

In this example, the page has a width of 9.90 cm and a height of 4.30 cm. The page contains six WordBox objects. Each WordBox object is populated with data from fields listed in the Data View.

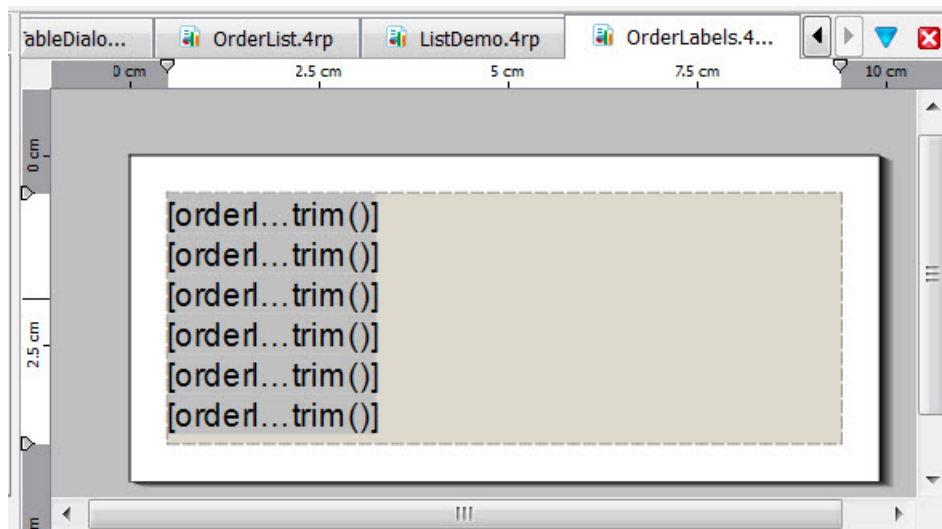


Figure 343: Label Report and Report Structure

5. In the **Report Structure**, place the report under the appropriate trigger.

In this example, the label (**Page**) is positioned under the **orderid** trigger, meaning a new label is printed each time there is a change in `orderid`.

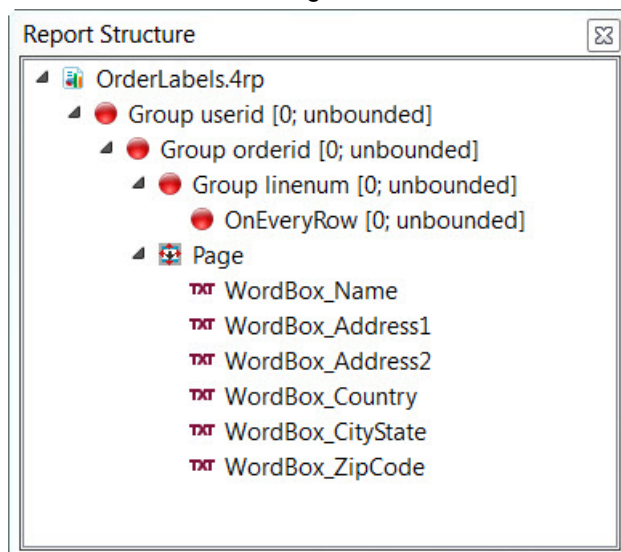


Figure 344: Label Report and Report Structure

6. Save your report.

For an example, see the `orderlabels.4rp` report in the OrderReport demo application.

Design address labels

Design for a label that may contain three to five lines, depending on the data record.

Before reading this procedure, you should be familiar with designing a basic label report. See [Design labels](#) on page 684 for more information.

A common label need is printing of address labels, yet the number of lines required for an address can vary, depending on the complexity of the address. In many database tables that store address data, there are several fields for storing address information, such as `addr1`, `addr2`, and so on. When creating an address record, those fields that are not needed for the new address are set to NULL.

When an address is printed out, however, those addresses that contain empty fields (`addr2` is set to NULL, for example) can cause an issue. No blank line should appear on the label. In addition, we may have information that we want to print after the last non-blank address line included (such as a postal code).

Follow this procedure to answer these address issues.

Note: Field names used in this example have been simplified. Use the full field names as they exist in the Data View.

1. Create your address label report.

- a) Use a **Vertical Box (Layout Node)** to contain all of the label data.
- b) Add all the lines of the address as children of this node, using dynamic layouting.

At this point, you have designed a report that prints an address label. If one of the lines is empty, however, a blank line is printed.

2. Identify which lines may contain empty values.

3. For each line that may contain an empty value, set the **visibilityCondition** to specify that the line not print if the content is blank.

For example, if one of the address label lines contains the data value `shipaddr2`, and this field has the potential of being empty, you could set the `visibilityCondition` as follows:

```
shipaddr2.trim().length()>0
```

With the `visibilityCondition` set properly, the line will not print if it has a length of zero. No blank lines appear within the address.

4. If you have a set of lines where some may be blank, and you wish to print something at the end of the last non-blank line, you set this up using a conditionality expression in the **value** property. With this expression, you test to see whether any of the subsequent (or following) lines contains a value. If one or more of the lines contains a value, the current line is printed. If non of the subsequent lines contain a value, then the postcode is appended to the end of the current line and printed.

For example, consider an address label containing three lines: `addr1`, `addr2`, and `addr3`. You have an additional field, `postcode`, that you wish to print after the last non-empty line.

- For the line containing `addr1`, we test and see whether `addr2` and `addr3` are empty by setting the value as follows: `addr3.trim().length()+addr2.trim().length()==0?addr1.trim()+postcode.trim():addr1.trim()`
- For the line containing `addr2`, we test and see whether `addr3` is empty by setting the value as follows: `addr3.trim().length()==0?addr2.trim()+postcode.trim():addr2.trim()`
- For `addr3`, it only prints if it is not empty (assuming the `visibilityCondition` is set correctly). Therefore, set the value as: `addr3.trim()+postcode.trim()`

With the `value` property set properly, the last non-empty line will have the postcode at the end.

Modify an object's borders, margins, or padding

Any box object on a report design document can have margins, borders, and padding.

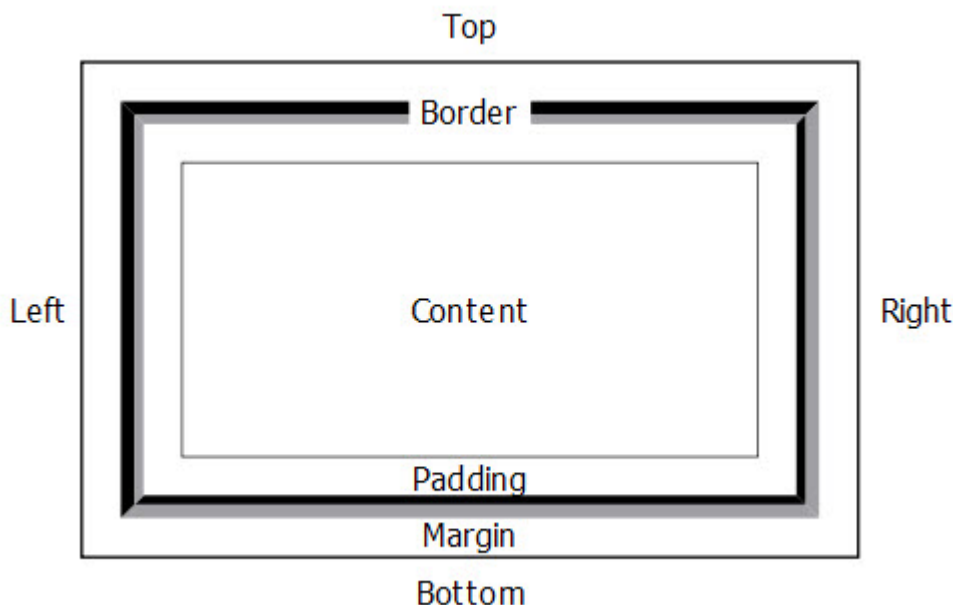


Figure 345: Border, Padding, and Margin

Set an object's [specific properties in the Properties View](#) to change:

- the width of a margin, border, or padding - [marginWidth](#), [marginRightWidth](#), [marginBottomWidth](#), [marginLeftWidth](#), [marginTopWidth](#), [borderWidth](#), [borderRightWidth](#), [borderBottomWidth](#), [borderLeftWidth](#), [borderTopWidth](#), [paddingWidth](#), [paddingRightWidth](#), [paddingBottomWidth](#), [paddingLeftWidth](#), [paddingTopWidth](#)
- the style of a border: solid, dashed, double, dotted, groove, ridge, inset, outset - [borderStyle](#), [borderRightStyle](#), [borderBottomStyle](#), [borderLeftStyle](#), [borderTopStyle](#)
- the color of a border - [borderColor](#), [borderRightColor](#), [borderBottomColor](#), [borderLeftColor](#), [borderTopColor](#)
- whether the box will have rounded corners (limited to the border styles solid, dashed, and double) - [roundedCorners](#)

Borders are **drawn outside the box** and will increase the actual size of the box beyond the value specified in x-Size and y-Size.

When a bordered item is positioned it behaves like a regular element, so that the attachment point appears at the specified position.

Illustrations

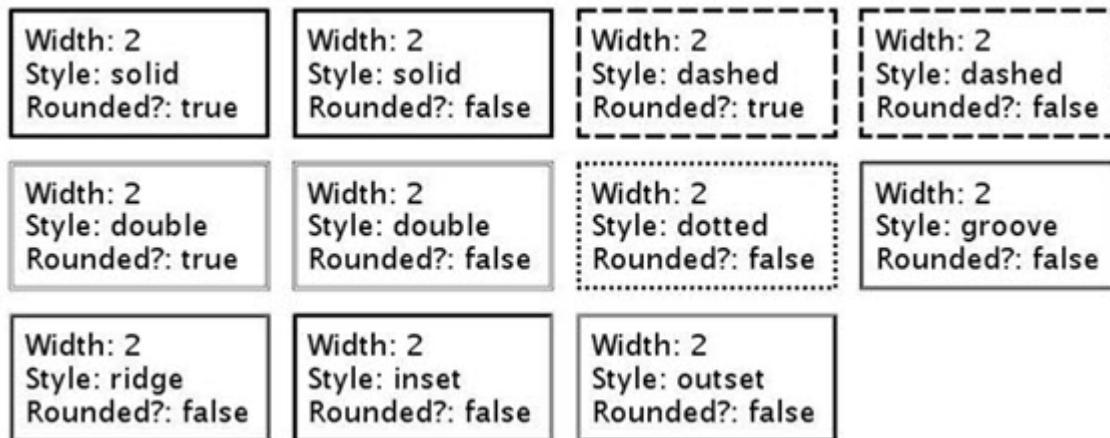


Figure 346: Examples of borders

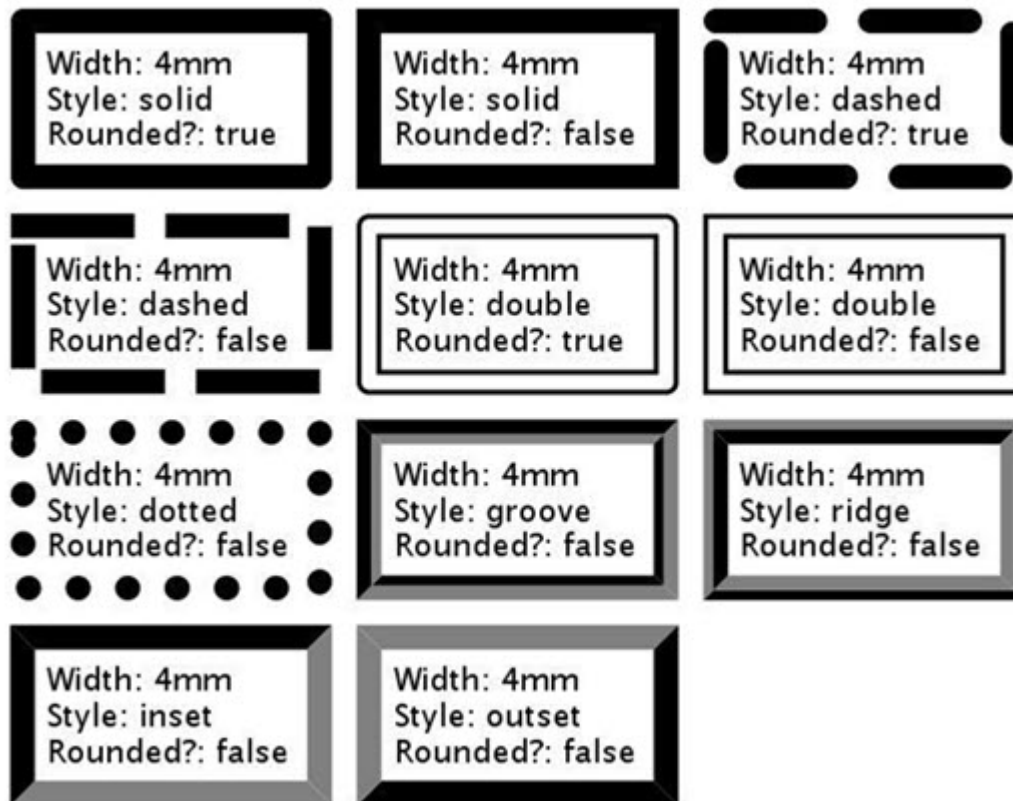


Figure 347: Examples of borders

Size expressions for bordered boxes
You can define the outer bounds of a box.

The [x-Size](#) and [y-Size](#) properties specify the inner size of the box; if we specify a box to be 3cm wide and have a 1mm thick border on all sides, for example, the box's outer bounds will appear to be 3.2cm wide. This conforms to the CSS specification.

You can define the outer bounds of a box instead:

- Determine the **x-Size** and **y-Size** values by subtracting the width of the borders from the desired height and width. For example, if you want a box to be 3cm wide on the outside while having 1mm borders on all sides, calculate the width to be $3\text{cm} - 2\text{mm} = 2.8\text{cm}$ wide.
- If you want a box to have the same size as its parent, however, set both the **x-Size** and **y-Size** properties to the value **max**. You do not have to subtract the borders, since the system automatically adjusts the value of **max** in cases where the box has borders. For example, if the box has a 1mm border and is contained in a box that is 3cm high and wide, the outer bounds of the contained box will also be 3cm.

Note: Do not use **expressions** that contain **max** as only one of its components, such as $\text{max}/2$, to specify the height and width of bordered boxes. Doing so can have unexpected results.

Size expressions that contain the variable **max** with other components

With bordered boxes, it may be necessary to customize expressions that use the **max** variable.

As explained in [Size expressions for bordered boxes](#) on page 688, the value of **max** is automatically adjusted by the border values. This causes unexpected results in expressions such as $\text{max} - 2\text{cm}$ or $\text{max}/2$, for example, where **max** is only a component of a more complex expression. Modify such expressions as follows:

1. Take the original formula and replace any occurrence of "max" with "(max+borders+padding+margin)" where "borders", "padding" and "margin" denote the width values for each on both sides of the box.
2. Take the resulting formula (which we'll call "f") from step 1, and create the final formula as "f-borders-padding-margin".

Example

LAYOUTNODE: x-Size="max/2", leftBorderWidth="2mm", leftMargin="1mm", rightBorderWidth="1.5mm", rightPadding="3mm"

Changing the expression for x-Size:

- Before Step 1: x-Size="max/2"
- After Step 1: x-Size="(max+2mm+1mm+1.5mm+3mm)/2"; this is "f" in the explanation.
- After Step 2: x-Size="(max+2mm+1mm+1.5mm+3mm)/2-2mm-1mm-1.5mm-3mm", which can be consolidated to "(max-7.5mm)/2-7.5mm".

The final property values for the box are:

LAYOUTNODE: x-Size="(max-7.5mm)/2-7.5mm", leftBorderWidth="2mm", leftMargin="1mm", rightBorderWidth="1.5mm", rightPadding="3mm"

Use hyperlinks in a report

Use the **id** and **href** properties to add hyperlink functionality to a report.

The **id** and **href** properties can be specified for text and images in the following containers: [wordBox](#), [wordWrapBox](#), [decimalFormatBox](#), [pageNoBox](#), [referenceBox](#), [imageBox](#), and [htmlBox](#).

id	Can be used to create an anchor in the document. Nodes can be identified with a unique id and then used as the target of a hyperlink.
href	Can be used to define a hyperlink pointing to any resource on the Internet, local disk, or any anchor

inside the document. The href should be defined using the URI syntax. For example:

```
http://www.google.com
```

```
mailto:santa.clauss@northpole.com
```

```
file:///C:/animals/images/honey_badger.jpg
```

```
#ref
```

Hyperlinks are not supported in reports output to Image, Printer or Postscript formats.

Some tips for legacy report designers

This table answers some common questions the correlation between Report Designer and traditional 4GL commands in reports:

Table 175: Legacy Report to Genero Report Writer

Legacy Report command	Using Genero Report Writer
SKIP TO TOP OF PAGE	In the report design document, drop a container that will consume the remainder of the page. See Forcing a page break .
BEFORE GROUP OF, AFTER GROUP OF	There is a GROUP trigger for data control breaks in the report structure. The position and contents of the child containers of the trigger determine what is printed out and when. See Group and Report totals .
ON EVERY ROW	In the ON EVERY ROW statement of the BDL file, the PRINT statement just sends the data items to the report engine. The report design document specifies what is to be printed out for every row of data passed to the report.
SPACES, format strings	All of the formatting for the report line is done in the report design document. These keywords are no longer used in a PRINT statement in the BDL file.
PRINT	In the report design document, the Data View tab displays the list of data items in the order in which they are specified in the PRINT statement.
ON LAST ROW	Drop a container positioned as the last child of the page root. The contents of the container will print out after the last report row. See Report Total .
NEED <i>n</i> LINES	Put all the report elements that need to be kept together in a Vertical Box Layout Node container. If there is not sufficient room on the page to print all of the elements in the container, the entire container will be printed on the next page.
PAGE HEADER, PAGE FOOTER	Avoid using these control breaks, which are triggered by the line count of the BDL report, which does not correspond with the actual page breaks

Legacy Report command	Using Genero Report Writer
	in the report output by Report Writer. Create page headers and footers in the report design document instead.

Backside printing

You can specify that something print on the back side of each report page.

An even page header (or footer) that uses all of the available space in layout direction (Y-size="max") will be printed "between" all pages.

If a report has the pages 1, 2, 3, 4 then the backside 'B' is printed between the pages resulting in the sequence 1, B, 2, B, 3, B, 4.

If a backside is required after the last page, then an additional backside page need to be defined at the end of the report yielding 1, B, 2, B, 3, B, 4, B.

Debugging your Report Design Document

Tips to help you debug issues you may have with your report design.

Using a Background Color

To check for overlap of an object on a report design, or to simply visually see where an object falls on your report, set the **Background Color** property of the object.

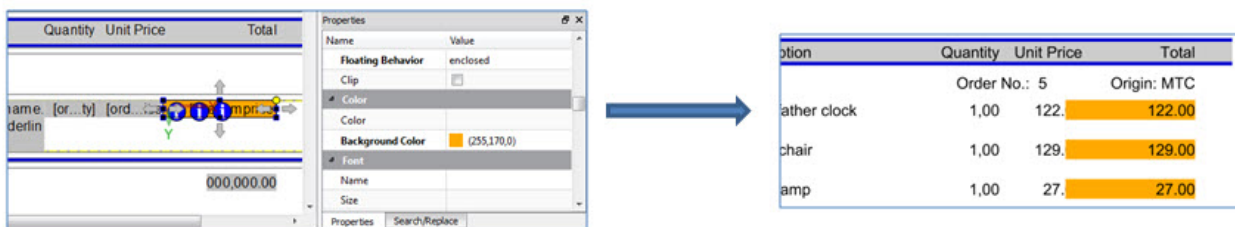


Figure 348: Setting the background color

Using GREDEBUG environment variable

Set GREDEBUG to check overfull boxes. Warning messages regarding any overfull boxes are written to standard output.

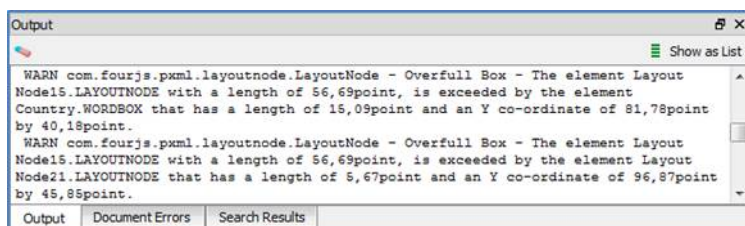


Figure 349: Overfull Box message

Working with tables

A table object has the ability to display data in columns and rows. You can add a table to a report, then manipulate the table to add or size columns and rows and change other display characteristics.

- [Add a table to a report](#) on page 692
- [Assign content to a table column](#) on page 692
- [Set the triggers for a table in a report](#) on page 692
- [Merge cells](#) on page 692

- [Add rows or columns](#) on page 693
- [Add headers and footers](#) on page 693
- [Change the width of a table](#) on page 693
- [Change the width of a column](#) on page 694

Add a table to a report

This procedure tells you how to quickly add a simple table to a new report.

While this procedure presumes you are starting with an empty report, it should provide you with the information you need to add a table to any report.

1. Create a new, empty report.
 - Select **File >> New, Reports, Empty Report (.4rp)** and click **OK**.
 - An empty report design document (4rp) displays.
2. In the **Data View**, open a schema file.
3. From the **Toolbox**, add a table to your report design document.
4. For each column, assign the field to display.
 - See [Assign content to a table column](#) on page 692.
5. Set the report trigger.
 - See [Set the triggers for a table in a report](#) on page 692.
6. Save and execute the report.
 - You will likely have to modify your report application to call your new report.

Assign content to a table column

A table is comprised of columns and rows. This procedure tells you how to associate a field from the Data View to a column header and body element.

Although this procedure tells you how to associate a field to a column, a column can contain any content - to include another table, a chart, any content.

1. Select the **Data View** tab. Identify the field you wish to use for a column in the table object.
2. Select the **Create an element based on the document context** icon from the **Data View** Toolbar.
3. Using relative positioning, drag the field from the Data View and drop it into the header row for the desired column.
 - It creates a column header. The column title is placed as a WordBox object. The Class property reflects that this is a title for the column, rather than a value.
4. Using relative positioning, drag the field from the Data View and drop it into the body row for the desired column.
 - It creates an expression with the value of the field. The field is placed as a drawable reflective of the data type of the field. The Class property reflects that this display the value (instead of the title).

Set the triggers for a table in a report

If you want each data row streamed to your report to result in a table row added to the table in your report, you need to set the appropriate trigger.

1. In the **Report Structure**, find the row element for your table.
2. Right-click the row element.
 - The context menu for the row element displays.
3. Select **Repeat selected items on >> On Every Row**.
 - When you place the table row under the **On Every Row** trigger, each data row results in one row added to the table.

Merge cells

With a row, you can merge one or more columns (cells) into a single cell. You can also revert the merge and convert the merged cell back to its original number of cells.

You can only merge the cells within a row. You cannot merge cells across rows.

1. Select multiple cells.

Hold down the Ctrl key and click on each cell you wish to merge. As each cell is selected, it turns blue in color.

2. Right-click and select **Merge Cells** from the context menu.

3. To reverse the merge, right-click on the merged cell and select **Split Cells** from the context menu.

The merged cell is split back into its original columns. Any content of the merged cell is put into the first column.

Add rows or columns

By default, a column has two rows and three columns. This procedure tells you how to add additional rows and columns.

To add a column or row to a report table object, you do not use the toolbox.

1. Right click on a row or column. either in the cell itself or on the control (selection tab) for the column or cell.

You will be inserting a row or column relative to the row or column you click. If you are inserting a new row, you will be able to add the new row above or below the selected row. If you are inserting a new column, you will be able to add the new column to the left or right of the existing column.

2. If you clicked on a cell instead of a control (selection tab):

- a) From the context menu, select **Insert Table Item**.

A second context menu displays with options to insert a row or a column.

- b) Choose the appropriate option to add a row or a column.

3. If you clicked on a column or row control instead of within a table cell, the context menu is specific to the column or row. Select the appropriate option from the context menu.

A new column or row is added to the existing table. Adding a column does not change the width of the table. The columns are resized to fit the new column.

Add headers and footers

By default, a table has a single header (Any Page Header). You can add additional headers or footers as needed.

To add a header or footer to a table, you do not use the Toolbox.

1. Right click on a cell in a row or column.

Which cell you click into is not relevant, as you will be specifying the type of header or footer to add.

2. From the context menu, select **Insert Table Item**.

A second context menu displays with options.

3. To add a header, select **Insert Header**. You are asked to choose between an any page header, a first page header, an even page header, or an odd page header. Select the appropriate header type.

If an option is grayed out, then that specific header has already been added to a report. To provide two rows for a specific header, you would not add two headers of the same type; you would add two rows to the specific header type.

4. To add a footer, select **Insert Footer**. You are asked to choose between an any page footer, a first page footer, an even page footer, or an odd page footer. Select the appropriate footer type.

If an option is grayed out, then that specific footer has already been added to a report. To provide two rows for a specific footer, you would not add two footers of the same type; you would add two rows to the specific footer type.

A new row is added to the existing table for the added header or footer. Headers are displayed at the top of the report design document, while footers are displayed at the bottom.

Change the width of a table

You can specify the width of a table. If you do not specify a width, the table expands to the width of the parent container.

This procedure assumes that you have not explicitly sized individual columns with an absolute value.

Note: You can set the width of a table, but you should not try to set the height of a table. The height is determined by the number of rows created.

1. Select the Table object.
2. Hover your cursor over the right-most border of the last column tab in the report until the resizing arrows appear.
3. Move the mouse until the table is the size you desire.
The `X-Size` property will change from the default (`max`) to a number representing the width you have selected.

Change the width of a column

If you do not specify a width, the columns are equal in width, and the width is calculated based on the width of the table itself.

A column width can be proportional or fixed.

When you specify a value in the Proportional Width property, you are specifying its width in proportion to other columns in the same table. Consider the following example: A table has three columns: A, B and C. Column A has a proportional width setting of 1, column B has a proportional width of 2, and column C has a proportional width of 3. This means that column B is two times as wide as column A, and column C is three times as wide as column A.

When you specify a value as Fix Width, you are giving an absolute size for that column. By default, the number entered refers to points, but you can change the unit of measure by specifying the type of units used. See [Unit Names](#) on page 801.

1. Select the table.
The controls (selection tabs) appear at the top of each column and to the left of each row.
2. Place the cursor on the right-side border of the selection tab for the column you wish to size. You can then drag the column to make it bigger or smaller.
3. For more precision, you can edit the property directly.
 - To change the column width in proportion to other columns, change the **Proportional Width** property. Using the up and down arrows on the property field increases and decreases in increments of one.
 - To specify a specific width, enter the value in the **Fix Width** property. Use the Reset button to clear the value from the **Proportional Width** property.

Working with business graphs

- [Report Writer Business Graphs](#)
 - [Map Chart](#)
 - [Category Chart](#)
 - [XY Chart](#)
- [Creating a Graph](#)
 - [The Data](#)
 - [The Design](#)
 - [A Custom Key](#)
- [Output Charts as Tables](#)

Report Writer business graphs

All of the Business Graphs in Genero Report Writer map Numeric values, grouping the data as required for the desired result.

Map Chart

The MapChart layout object allows you to create a graph that has one set for values to be mapped, grouped together by a specific key.

The MapChart layout object is defined by the [MapChartDrawAs](#) class.

Draw As property: This kind of graph can be drawn as a Pie, Pie3D, Bar, Bar3D or Ring, or as a [table](#). For example, the pie chart in [Figure 350: Pie Chart example](#) on page 695 presents the total revenue (the value) for each customer (the key), based on the `OrderReport.rdd` file in the Report Writer sample programs.

Draw Legend and **Draw Labels** property: These properties have been added to customize the appearance of the plots. In particular the option to remove the legend is useful when more than several charts are drawn next to each other in a document; the option can be used to make the charts share a single legend by specifying the legend only on one of the charts.

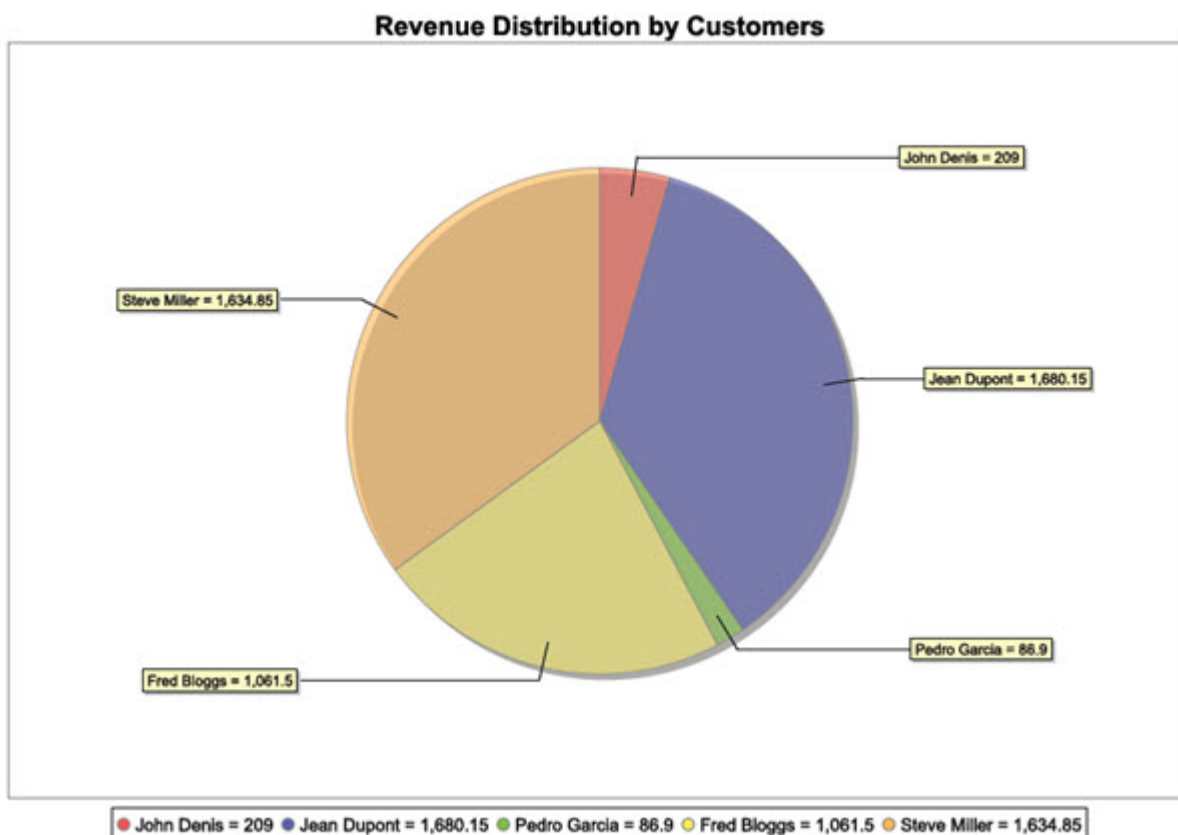


Figure 350: Pie Chart example

The **Map Chart Item** layout object specifies the key data item and the values data item in its properties:

- **Key** - the data is to be grouped by this property (customer name in the example chart); must be a String.
- **Value** - the chart will display the total of this property (total unitprice in the example chart) for each key (customer name); must be Numeric.
- **Name** - name of this report item in the [Structure view](#); must be a String.
- **Color** - gives each slice a specific color. When a color is specified for a particular key in one chart, then the same color will be used for that key in other charts too, unless specified otherwise. If different colors are specified for the same key, the most recent value is used. If the same color is specified for a number of different keys, only one of these keys will be painted with the specified value; the other slices will be painted with interpolated values. Charts may use gradients, shading, or translucency with the colors specified.

Category Chart

The Category Chart layout object allows you to create a graph that has two keys for each set of values.

Draw As property: This kind of graph can be drawn as a Bar, Bar3D, Stacked Bar, Line, Line3D, Area, Stacked Area, Waterfall, or as a [table](#).

In a waterfall graph, the value in the last category of the data set should be (redundantly) specified as the sum of the items in the preceding categories - otherwise, the final bar in the chart will be incorrectly plotted. At the present time, the waterfall graph can only have one category.

This Bar chart presents total revenue (the value) by the Customer Name and Product Category (the keys):

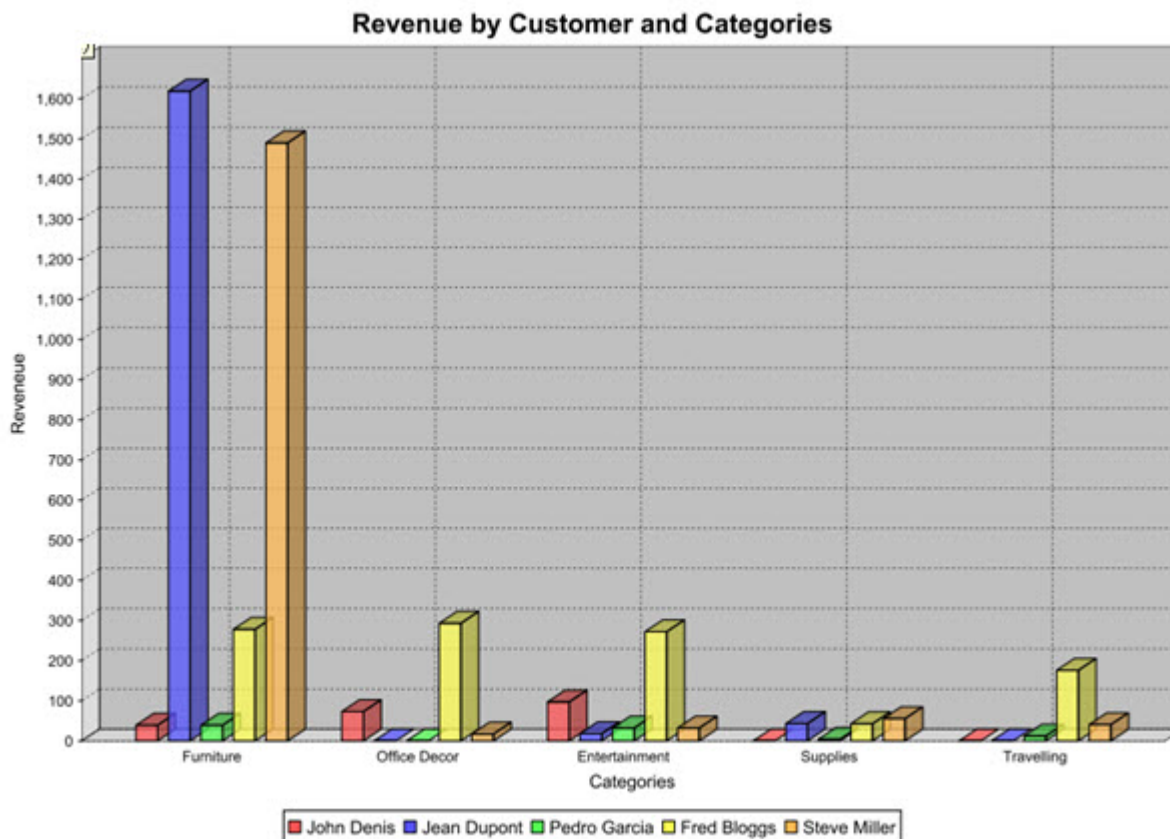


Figure 351: 3-D Bar Chart example: Revenue by Customer and Categories

The **Category Chart Item** layout object specifies the key data items and the values data item in its properties:

- **Category Key** - the categories for the data (the type of product purchased in the example chart); must be a String.
- **Key** - the data in each category is grouped by this property (the customer name); must be a String.
- **Value** - the chart will display the total of this property (the unitprice of order line items) for each category (product) divided into groups by Key (customer name); must be Numeric.
- **Name** - name of this report item in the **Structure view**; must be a String.

XY Chart

The XY Chart layout object allows you to create a graph that maps a series that has two values, as an XY-plot. This chart is used most often for scientific data.

Draw As property: This type of graph can be drawn as a Polar, Scatter, Area, Stacked Area, Line, Step, Step Area, a [table or sorted table](#), or Time Series (property.) This graph presents the x and y values of three series of trigonometric functions.

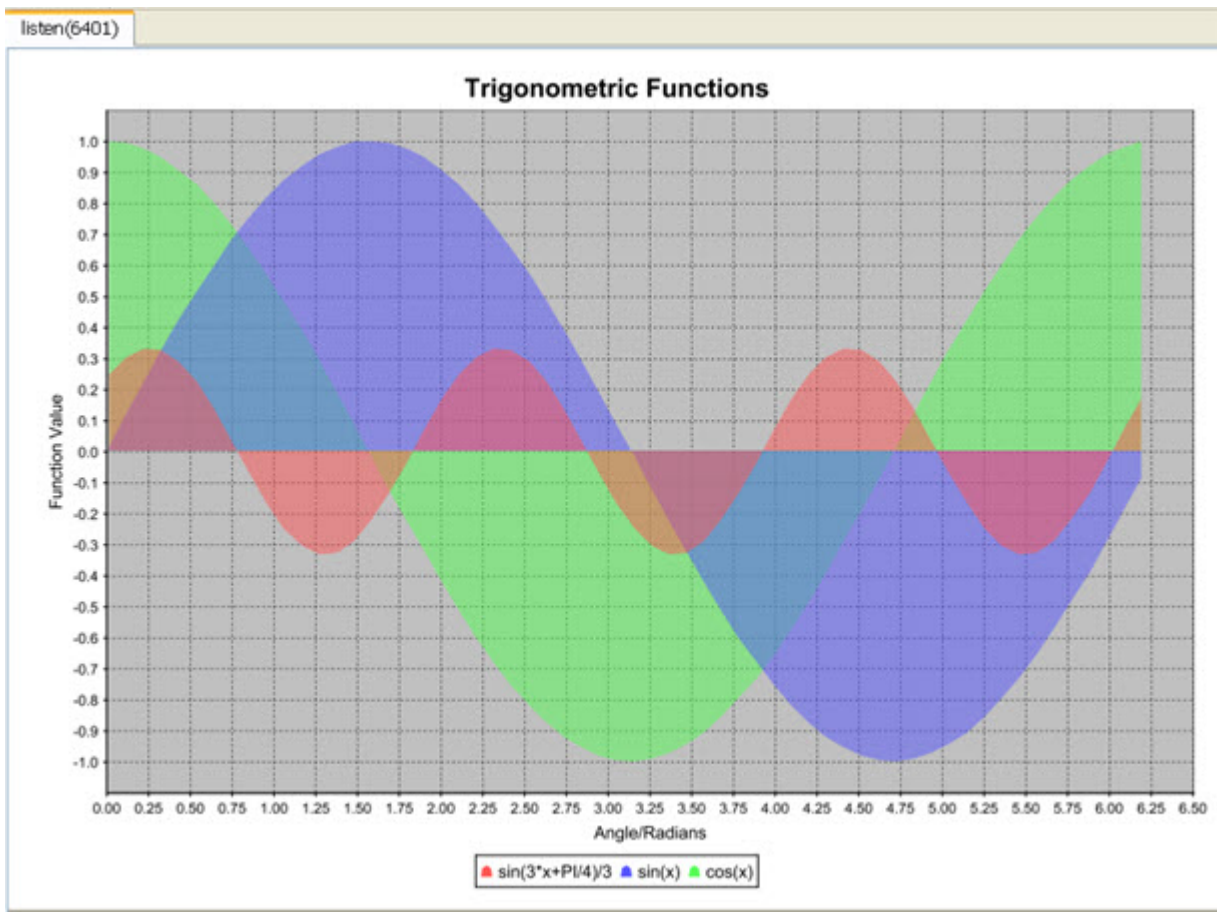


Figure 352: XY Chart example

The **XY Chart Item** element specifies the value data items in its properties:

- **Series Title** - the caption for the series of values being charted; there can be more than one series (three in [Figure 352: XY Chart example](#) on page 697); must be a String.
- **X** - the values for the x axis; must be Numeric.
- **Y** - the values for the y axis; must be Numeric.
- **Name** - name of this report item in the [Structure view](#); must be a String.

Creating a graph

A Business Graph uses the same process as any other type of report.

See [Steps to a Report](#) for a list of the steps required to create a report.

The data

As with all reports, the data schema defines the data for the graph.

As in other types of reports, the PRINT statement in the BDL file specifies the data structure of the information that is available for the graph:

```
ON EVERY ROW
LET lineitemprice = orderline.lineitem.unitprice *
orderline.lineitem.quantity
LET overalltotal = overalltotal + lineitemprice
LET ordertotal = ordertotal + lineitemprice
PRINTX orderline.*, lineitemprice, overalltotal, ordertotal
```

See [Creating the BDL File](#) for additional information.

You extract the data structure as an `rdd` file that you reference in the Data View page of the Report Designer:

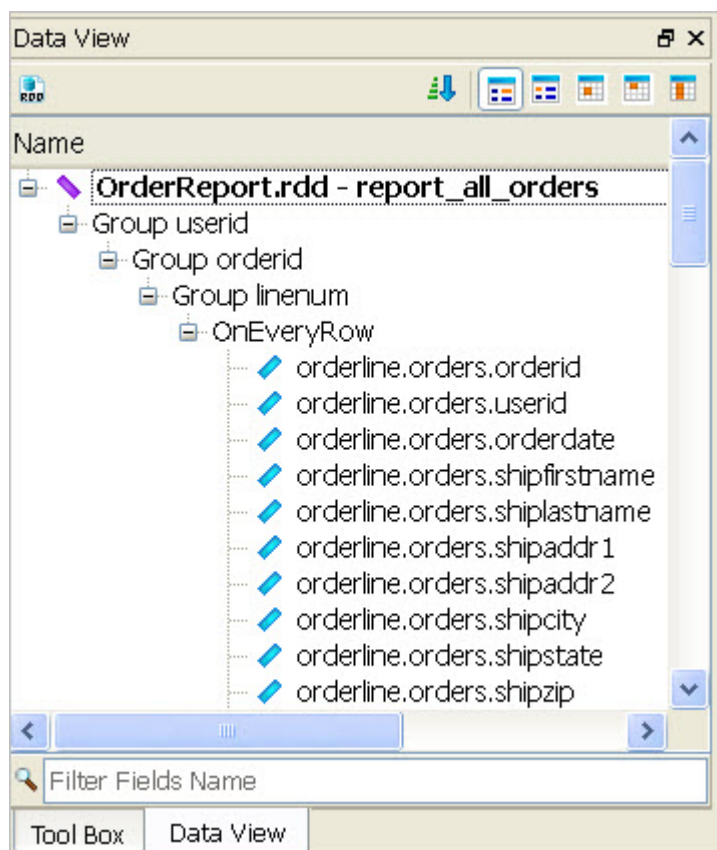


Figure 353: Report Designer Data View

A single `rdd` file can be used to design multiple reports containing the values of various data items from the file. The same data items can also be displayed as different types of charts.

The design

1. Open a new or existing report in Report Designer.
2. If you are creating a new report, [specify the data schema](#) to be used for the report.
3. From the Tool Box view, drag and drop the desired Business Graph (Map Chart, Category Chart, XY Chart, or Pivot Table) into a container on the report design.

The Design Window will contain a Chart object and its related Item object (shown as a price tag). If you cannot view the entire chart in the Report Design window, you can re-size the display of the report in the window: point the mouse at the report and scroll the mouse scroll button while holding down the keyboard Control key.

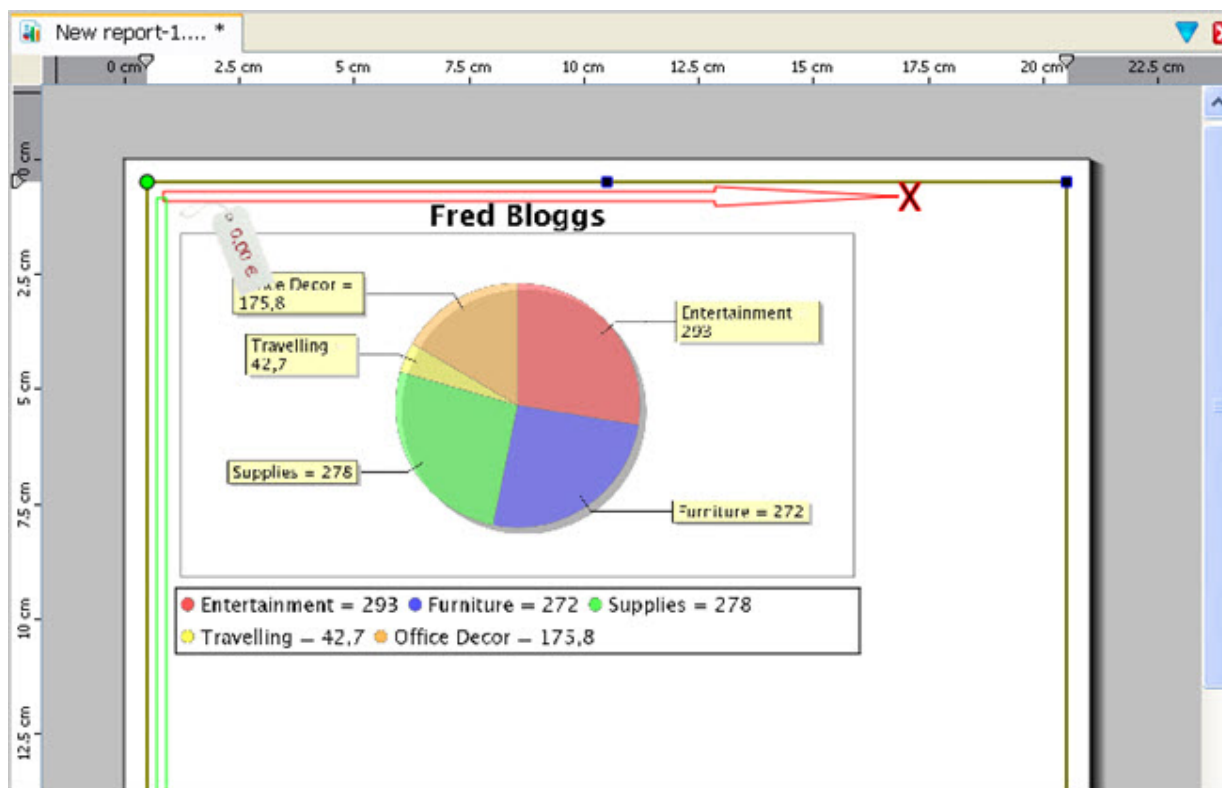


Figure 354: Chart object in the work area

4. Select the chart object and enter the values for its properties in the [Properties View](#). The properties vary depending on the graph chosen:
 - [Title](#) - caption at the top of the graph
 - [Values Title](#) - caption for the values
 - [Keys Title](#) - caption for the keys
 - [Category Title](#) - caption for the categories (of a [Category Chart](#))
 - [X Axis Title](#) - caption for the X axis (of an [XY Chart](#))
 - [Y Axis Title](#) - caption for the Y axis (of an [XY Chart](#))
 - [Draw As](#) - the type of chart
 - [Sort By](#) - sort alphabetically, numerically, or by input order.
 - [Sort Ascending](#) - sort in ascending or descending order.
5. Select the chart's price tag, which represents the item object, and enter values for its properties. The specific properties vary depending on the graph type.
 - **keys** - the data items that are used to group values, These data items must be Strings. You are not limited to existing String data items, however. You can define a [custom string for the key](#), using the data items in an expression.
 - **values** - the data item that contains the numbers to be charted. These data items must be Numeric.
6. Modify the tree in the Structure view as needed:
 - The item object for the chart ([Map Chart Item](#), [Category Chart item](#), [XY Chart item](#)) must be a child of the [OnEveryRow trigger](#) node.
 - The location of the chart object ([Map Chart](#), [Category Chart](#), [XY Chart](#)) specifies how many different charts will be created, based on the resulting subsets of the data.

Drag and drop the nodes in the [Structure view](#) to create the correct hierarchy. If you drop a container or trigger on a different node, it will be a child of that node. Use alt-drag and it will become the parent of that node.

Examples

With [Figure 355: One chart per each row](#) on page 700, the Structure View that resulted from the creation of a new report containing a chart. It will create a new page and chart for each data row passed to the report, and therefore needs to be modified:

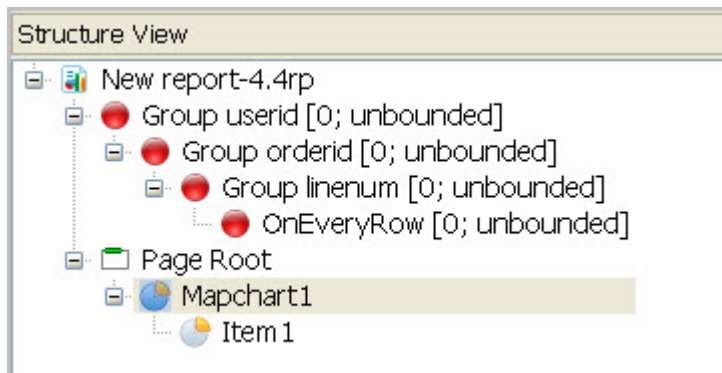


Figure 355: One chart per each row

With [Figure 356: One chart, one page](#) on page 700, the Structure view will result in a page containing one chart; the item node is a child of OnEveryRow:

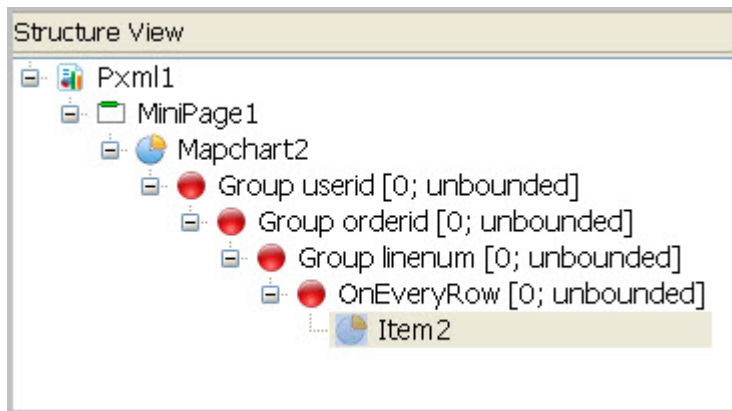


Figure 356: One chart, one page

With [Figure 357: One chart for each unique userid](#) on page 700, the Structure View will result in one chart for every unique userid, since the Map Chart is a child of the userid trigger node; the item node is a child of OnEveryRow:

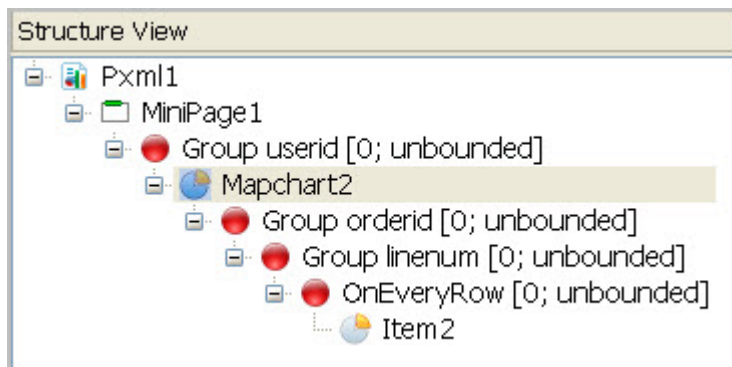


Figure 357: One chart for each unique userid

Custom keys

You can enter any valid expression for the String value of a key property.

This could be a substring or a concatenation of existing Strings. For example, this expression would group the data values based on the first letter of shiplastname:

```
orderline.orders.shiplastname.substring(0,1)
```

Examples

This chart uses the last name as the key (trimmed of trailing blanks), as shown in the Properties view. The unit price on each order for each unique last name is rolled up to a total as shown on the chart. There are five unique names:

Properties	
Name	Value
Items	
Key	{orderline.orders.shiplastname.trim()}
Value	{orderline.lineitem.unitprice}
Name	Item2

Figure 358: Properties panel for the chart Items (where key uses trimmed last name)

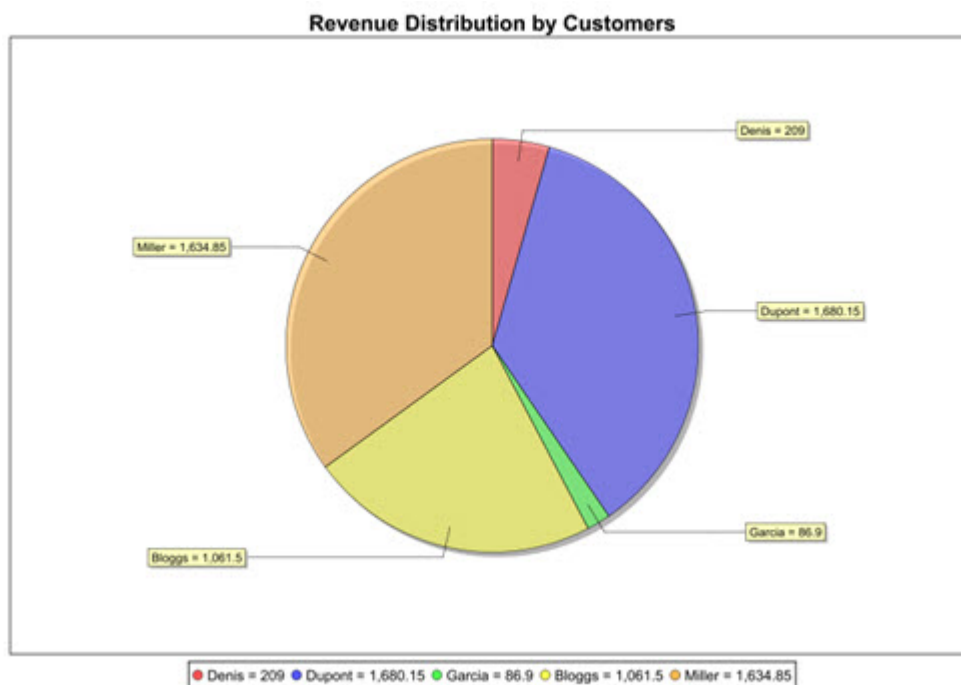


Figure 359: Chart showing Revenue Distribution by Customers (where key uses trimmed last name)

And this chart uses the first letter of the last name as the key, as shown in the Properties View. The unit price on each order for each unique first letter is rolled up to a total as shown on the chart. In this chart, there are only four unique first letters, as two customers have last names beginning with D:

Properties	
Name	Value
Items	
Key	orderid.orders.shiplastname.substring(0,1)
Value	{orderid.lineitem.unitprice}
Name	Item2

Figure 360: Properties panel for the chart Items (where key uses substring)

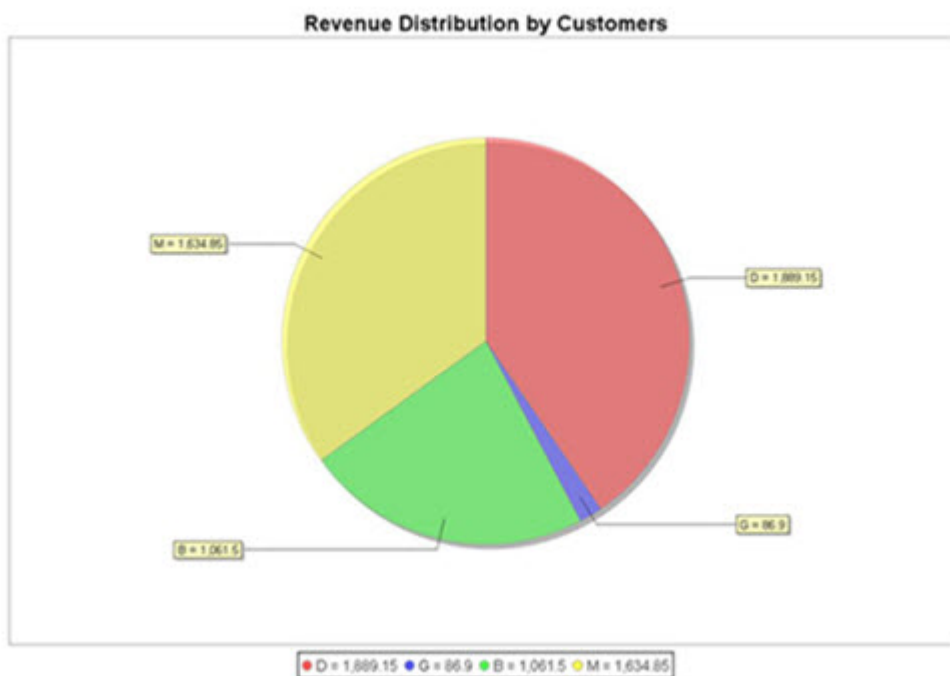


Figure 361: Revenue Distribution by Customers (where key uses substring)

Output charts as tables

All chart types now support the drawing of the data as tables via the Draw As property.

The tables drawn are "pivot" tables that may contain subtotals. This terminology and rules are applied:

- The columns of the tables are either of type **Dimension** (the data items that are used to group values) or of type **Value** (see [Mapping of Chart properties](#)).
- **Values** are aggregated or totaled by the **Dimension**.
- **Dimension** columns precede **Value** columns in the table.
- The order of the **Dimensions** specifies the order of the data.
- Subtotals are generated for each **Dimension** except the rightmost.

Mapping of chart properties

- **Map chart** - the **Key** property is mapped to a **Dimension** column, and the **Value** property to a **Value** column.
- **Category chart** - the **Category Key** property and the key property are mapped to **Dimension** columns, and the **value** property to a **Value** column.
- **XY chart** - the **Series Title** property is mapped to a Dimension column and the **X** and the **Y** properties are mapped to Value columns.

There are three types of tables: Table, Sorted table, and Aggregated table.

Table

This option lists all data items in a table. The data needs to be presorted in the order of the dimensions; if this is not the case, the table will contain useless subtotal rows.

If the number of rows in the table is large, then **Table** is the preferred choice since it produces the tabular output row by row while reading the input and does not keep a copy of the table data in memory. In other words, this option does not delay the output until the end of the input has been read. as the Sorted Table and Aggregated Table options do.

Sorted Table

This option produces the same output as the Table option, but the data does not need to be presorted. The output is delayed until the last row of table data has been read, and the entire table data is stored in main memory.

Aggregated Table

This option draws the same table as the previous two options, but subsequent rows with identical dimensions are drawn only once and the total values are computed. This option always sorts the data, and delays the output until the last row of data has been read. This option is not available for [XY chart](#) types.

Working with Pivot Tables

The pivot table element defines a table element with fixed roles and types for its columns. Grouping, sorting and summarizing allows results to be displayed in different ways.

- [What are pivot tables?](#) on page 703
- [Sample pivot table reports](#) on page 706
- [Create a static pivot table](#) on page 707
- [Pivot table properties](#) on page 708
- [Arrange your hierarchies](#) on page 708
- [Add a dimension](#) on page 709
- [Add a measure](#) on page 709
- [Pivot table elements and the Structure view](#) on page 710

What are pivot tables?

The pivot table element is a table element with fixed roles and types for its columns, suitable for processing multi-dimensional data. Grouping, sorting, and summarizing are performed. The results can be displayed in different ways.

A pivot table has two types of columns: *dimensions* and *measures*. A column is either a dimension or a measure. No other column types exist in a pivot table. A pivot table has one type of row: *fact rows*. The values in the cells of a row are either *dimension values* or *measures*, depending on the column type.

Data is sorted by the dimension values. There are usually many rows with identical dimension values in a column. The dimensions can be viewed as forming a hierarchy. For this reason dimension can also referred to a *hierarchies*.

A measure is aggregated. If the measure is numeric, the aggregation could be an average of the measure values, the sum of the measure values, the maximum or minimum of the measure values, and so on.

For example, consider a table with the dimension columns "Country" and "Region". After sorting the data, several rows starting with {"Afghanistan", "1 North", ..} will be at the top, perhaps followed by some rows starting with {"Afghanistan", "3 South", ..} again followed by rows starting with {"Albania", "1 North", ..}. "Country" and "Region" form a hierarchy or tree where a country branch has sub branches for it's regions. The innermost dimension is said to contain the "facts" or "values" (meaning the measure columns from the fact rows). In a tree representation, the leaves of the tree are records containing the values for the measure columns.

Relationship to charts

The pivot table is a generalization the chart objects. As an example one can say that a CATEGORYCHART is a PIVOTTABLE with two dimensions (The "categoryKey" and "key" attributes in the CATEGORYITEM element), one measure (the "value" attribute in the CATEGORYITEM element) on which a summarizing aggregation is performed for both dimensions. This table compares the different chart objects to the pivot table.

Table 176: Comparing chart objects and pivot tables

Element type	Number of dimensions	Number of measures	Number of aggregation groups	Aggregation functions	Sorting options
MAPCHART	One (specified by the key attribute)	One (specified by the value attribute)	One (values with the same key value are summarized)	Summarizing	By key, value and input order
CATEGORY CHART	Two (specified by the key and categoryKey attributes)	One (specified by the value attribute)	One (values with the same key + categoryKey value combination are summarized)	Summarizing	By keys, value and input order
XYCHART	None	Two (specified by the x and y attributes)	None	None	None
PIVOTTABLE	N (specified by HIERARCHY elements)	N (specified by MEASURE elements)	N (Aggregation can be performed on all dimensions)	Summarizing and others (such as count, average, maximum, minimum, and so on)	Input order and any combination of measures

Set data types

The pivot table makes use of the set data types Column Selector and Order Specifier.

These set data types are defined:

Column Selector

The column selector is a comma-separated list of positive integers numbering 0 for the first column and n-1 for the last column in a table of n columns. Example: Given a table with four columns, a column selection of "0,3,1" selects the first, the last and the second column.

Order Specifier

The order specifier is a comma-separated list of positive integers numbering 0 for the first column and n-1 for the last column in a table of n columns. Positive values specify ascending, negative values descending order.

Example: Given a table with four columns, a order specifier of "-0,3" specified an descending order on

the first column and a ascending order on the last column.

There is a difference between not setting a value and specifying an empty value. The empty value always means the empty set. Not setting a value may mean selecting all or nothing, depending on the context.

Runtime configurability

The number of possible visualizations for a single pivot table data model is huge.

Consider a table with the dimensions “Country”, “Salesperson” and “Year” and the measure “Turnover”.

The list of possible views:

- Total Turnover by Country
- Total Turnover by Country and Salesperson
- Total Turnover by Country, Salesperson and Year
- Total Turnover by Country and Year
- Total Turnover by Country, Year and Salesperson
- Total Turnover by Salesperson
- Total Turnover by Salesperson and Year
- Total Turnover by Salesperson, Year and Country
- Total Turnover by Salesperson and Country
- Total Turnover by Salesperson, Country and Year
- Total Turnover by Year
- Total Turnover by Year and Country
- Total Turnover by Year, Country and Salesperson
- Total Turnover by Year and Salesperson
- Total Turnover by Year, Salesperson and Country

Adding one more dimension multiplies the number of variations by more than four.

If we add two more measures “Margin” and “Cost”, the number of variations is multiplied by two as we can now view “Total Margin by ...” and “Total Cost by ..” for all existing variations.

If we compute not only the total but also the average and the maximum, the number is again multiplied by two, as we can now view “Average Turnover by ..” and “Maximum Turnover by ..”

If we sort the output by some measure and display only the top n items, the number of options is again multiplied by the number of measures and n, since we can produce view such as “Top 3 Selling Countries” or “Top 5 Salespersons regarding margin”.

The number of variants could be multiplied by the 20+ drawing options (specified by the drawAs property).

Instead of having to provide a separate `4xp` file for each variant, the implementation of pivot tables allows the creation of pivot table models containing a larger amount of dimensions and measures which will likely never be displayed as a whole. From the static setup, one can then select dimensions and measures for display via selection properties. By defining RTL expression for these properties, one can create the different variants described at runtime.

Performance considerations

A pivot table report is capable of handling large amounts of data without exhausting memory, as long as some constraints are met.

If tabular output is selected and other constraints are met, output is produced without delay and memory consumption is nearly constant. The processing time is proportional to the input length; for very large data sets it is advisable to aggregate the data in the database.

Processing should be latency free

A chart displays on a single page. As such, it displays only after all data has been processed.

When outputting a table, the output can span multiple pages. Data can be output during processing, a page can be returned well before all data is processed. Yet selecting this visualization type alone does not ensure latency free processing; the data must be pre-sorted (See the [hierarchiesInputOrder](#) property). If the data is partially sorted, there can be periods of delay while the processor waits for the end of a block of data that needs to be sorted.

Pre-sorting data reduces memory consumption

Sorting is done in memory. Very large reports should therefore be run on (partially) pre-sorted data (See the [hierarchiesInputOrder](#) property). Output sorting is also done in memory (See the [outputOrder](#) property) and should be used with equal care. Suppressing the display of the fact rows (See the [displayFactRows](#) property) can significantly reduce memory consumption.

Not sending duplicate values reduces processing time

In the case that data is pre-sorted (see the [hierarchiesInputOrder](#) property), an optional, more compact form of data representation can be chosen that allows omitting dimension values that did not change from one row to another, thereby improving performance.

For example, after shipping the first fact row {"Afghanistan","1 North",...} all subsequent rows that contain measure for north Afghanistan need not ship these two dimensions anymore. When the first row of the next block {"Afghanistan","3 South",...} is reached only the value "3 South" needs to be reported once on the first row of the block. See [Pivot Table Hierarchy Value](#) on page 733.

Sample pivot table reports

Two sample pivot table reports are provided with the installation of Genero Studio. One report shows a static pivot table, while the other shows a dynamic pivot table.

The reports are located in the `Reports.4pw` project. This project can be found in the `My Genero Files/samples/Reports` directory.

Static pivot table sample report

The report name is `StaticPivotTable.4rp`.

This sample report produces a table of customer data, grouped by customers and orders. The input is presorted. The dimension columns, the `userid` and `orderid`, are populated accordingly.

Dynamic pivot table sample report

The report name is `DynamicPivotTable.4rp`.

When this report is selected, a second dialog opens. From this dialog, you select the dimensions and measures included in the report, along with how to sort the measures.

The form `pivotdialog.4fd` defines the dialog. The module `pivotdialog.4gl` contains the Genero code driving the dialog, making use of the [pivot table library `libpivot.4gl`](#).

A new record of type "PivotControlBlock" is shipped in the FIRST PAGE HEADER of the report. The structure of the record:

```
TYPE PivotControlBlock RECORD
  title STRING,
  drawAs STRING,
  dimensionsDisplaySelection STRING,
  measuresDisplaySelection STRING,
  outputOrder STRING,
  topN INTEGER,
  displayFactRows BOOLEAN,
  displayRecurringValues BOOLEAN,
  computeAggregatesOnInnermostDimension BOOLEAN,
  computeTotal BOOLEAN,
```

```

computeCount BOOLEAN,
computeDistinctCount BOOLEAN,
computeAverage BOOLEAN,
computeMinimum BOOLEAN,
computeMaximum BOOLEAN
END RECORD

```

The definition of the record is located in the file `globals.4gl`.

The record is populated by a call to the dialog function `promptForPivotDialogIfAny` (from `pivotdialog.4gl`) in `OrderReport.4gl` in the sample application. This is the code fragment:

```

CALL promptForPivotDialogIfAny(filename) RETURNING retval, controlBlock.*

IF NOT retval THEN
    RETURN NULL
END IF

IF NOT fgl_report_loadCurrentSettings(filename) THEN
    EXIT PROGRAM
END IF

```

The function inspects the `4rp` file. If it finds exactly one dynamically configurable pivot table, it will prompt the user to configure it (see `pivotdialog.4gl` for details).

The last step lies in `DynamicPivotTable.4rp` where the pivot table properties are defined as RTL expressions that initialize from the field values in control record. For example, the **title** property is initialized to `"controlBlock.title"`.

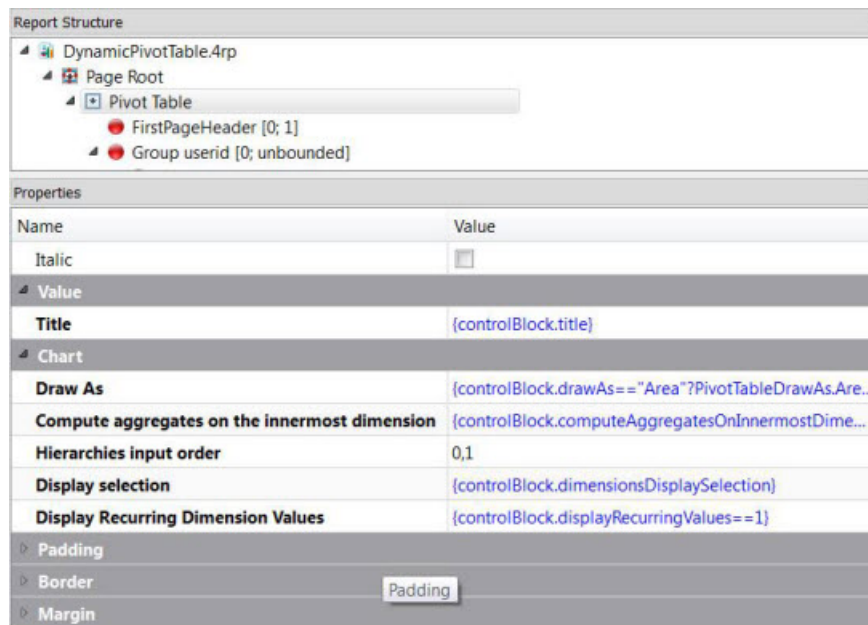


Figure 362: Properties of the Pivot Table element

This figure displays the values of the properties for the pivot table element.

Create a static pivot table

Follow this procedure to create a static pivot table.

1. Create a new, empty report.
2. In the Data View, associate your data schema.
3. From the Tool Box, add a Pivot Table to the report layout under the Page Root.
4. Add dimensions as Hierarchy elements. See [Add a dimension](#) on page 709

5. Add measures under the Fact node. See [Add a measure](#) on page 709.
6. Arrange the dimensions and measures in the Structure View. See [Arrange your hierarchies](#) on page 708.
7. Set additional properties as needed for all elements of the Pivot Table. See [Pivot table properties](#) on page 708

An example is provided in the Reports project (`Reports.4pw`), provided as part of the `samples` directory. The report is `StaticPivotTable.4rp`.

Pivot table properties

When you define a pivot table, you set properties specific to the pivot table.

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

In addition to the attributes available for LAYOUTNODE, the PIVOTTABLE element includes the following properties:

- [title](#)
- [drawAs](#)

The `drawAs` property specifies the type of output that is rendered from the data. Depending on the type selected and the number of available dimensions, the rendering is delegated to the map chart, category chart, XY chart or table element. In case that the number of selected dimensions outnumbers the respective number in the selected visualization, the exceeding dimensions and measures are ignored. The values are assigned from left to right so that if for example a pivot table with 4 dimensions and 3 measures is drawn as a category chart which has only 2 dimensions and one measure, then the chart will be drawn using the first two dimensions and the first measure from the pivot table's columns. Selecting "Table" causes the output to be drawn in tabular form, displaying all selected columns of the pivot table.

Valid values for Pivot Table: Pie, Pie3D, Ring, Bar, Bar3D, Area, StackedBar, StackedArea, Line, Line3D, Waterfall, Polar, Scatter, XYArea, XYStackedArea, XYLine, Step, StepArea, TimeSeries, Table

Arrange your hierarchies

In order to minimize the volume of the data stream, the HIERARCHIES can be shipped sparsely, in the sense that not all hierarchy values need to be shipped for every row.

Comparing the hierarchy values of two consecutive rows from right to left starting with the innermost dimension, any value can be omitted that is the same in both rows until reaching the first dissimilar column.

This example shows two equivalent pivot tables. The first table uses a flat representation while the second ships the values using a more space efficient method. It ships the value for "userid" only on changes of "userid" by placing the value in the corresponding trigger. The value remains the same for all orders of that customer. The same is done for "orderid" which changes its value for every order but remains the same for all items within the order. The arrows in the diagrams indicates the location of the hierarchies.

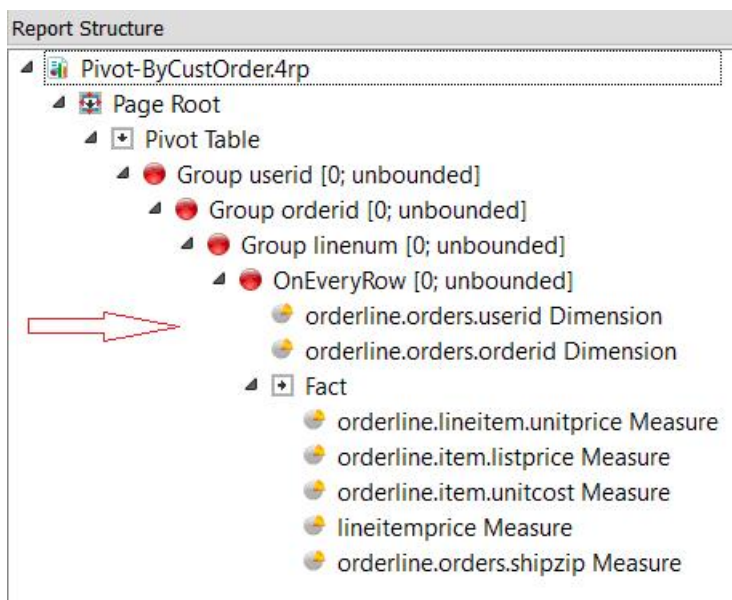


Figure 363: Flat shipping of dimension values

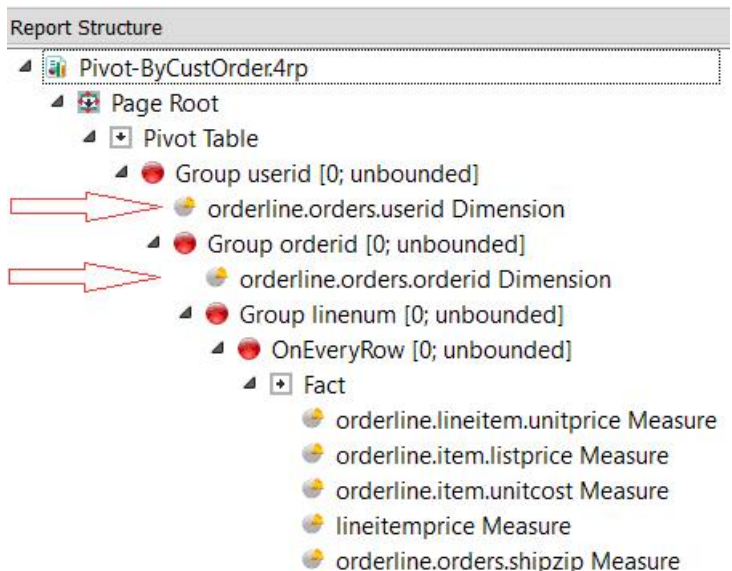


Figure 364: Optimized shipping of dimension values

Add a dimension

Dimensions are the values against which aggregation occurs.

- In the Value property, specify the column name for the dimension.
- Select the desired aggregation types. Compute Totals is selected by default.
- In the Title property, provide a title for the dimension.

Add a measure

Measures are placed under a Fact in the Structure View.

- In the Value property, specify the column name for the measure.
- If numeric, check the Numeric Column checkbox.
- In the Title property, provide a title for the dimension.
- If numeric, specify a format.

Pivot table elements and the Structure view

A static pivot table uses predefined dimensions and measures when creating the report.

A pivot table is constructed from four elements.

- The PIVOTTABLE element represents the table itself and all other elements are contained in it.
- The columns of the table are described by HIERARCHY elements (in case of a dimension column) or MEASURE elements (in case of a value column).
- MEASURE element are grouped in FACT elements.

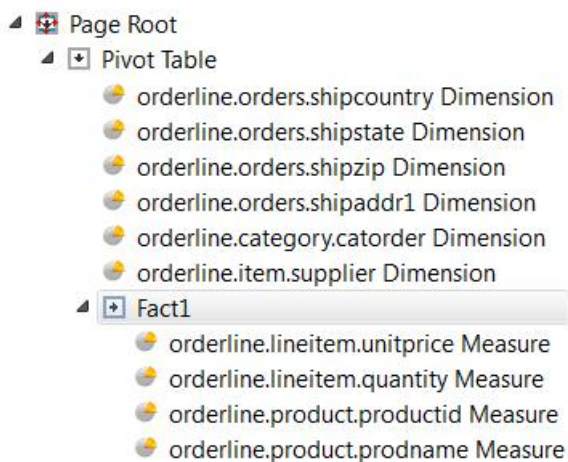


Figure 365: Pivot table elements in the Structure View

This image shows a table with six dimensions (HIERARCHY elements) and four measures (MEASURE elements).

Column definition

This figure shows a pivot table definition on the left and a possible rendering of the table on the right.

- Dimensions and measures define the columns of the table.
- Not all defined dimensions and measures were selected for display.
- The title of the table and the selection of the dimensions is defined in the pivot table element
- The title of the columns are defined in the dimension and measure elements.
- The selection of the measures is defined in the fact element.

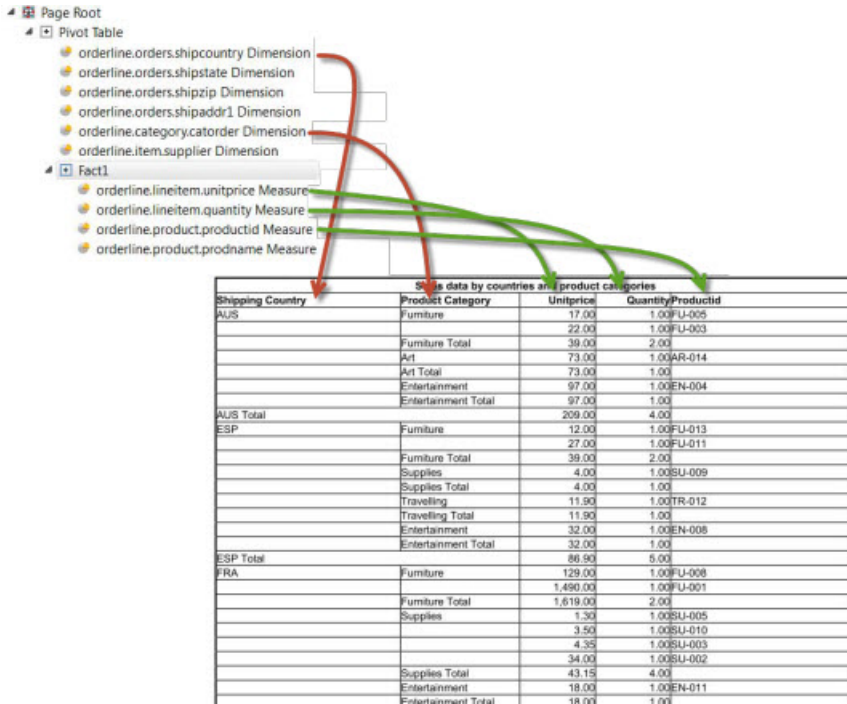


Figure 366: Report columns

Row definition

The entity of dimension declaration followed by one fact element forms a row of a table. Typically one row is defined. It is placed in a trigger to get repeated for each input record.

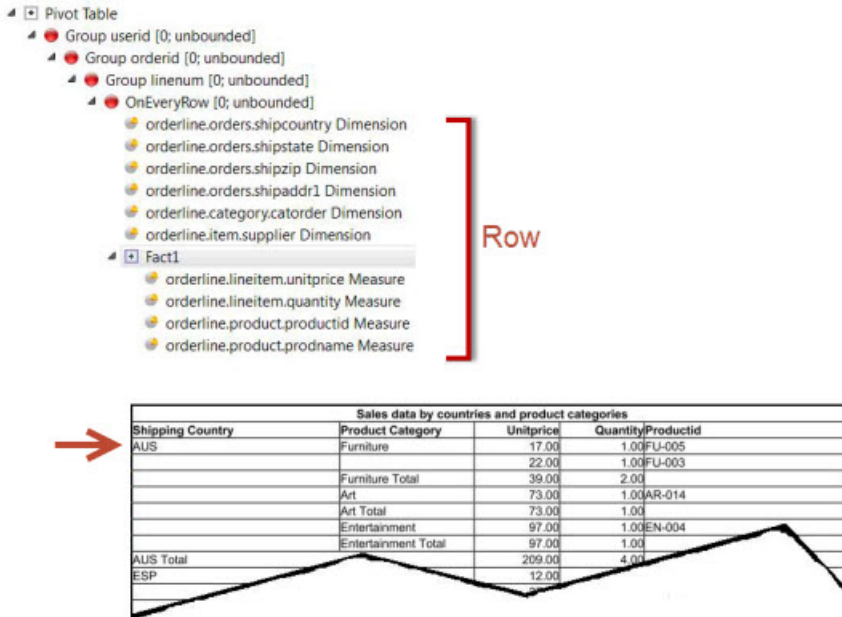


Figure 367: Report row placed in a single trigger

It is allowed, but highly unusual, to specify the rows literally. All rows have to have exactly the same structure (number of dimensions and measures, types, and so on).

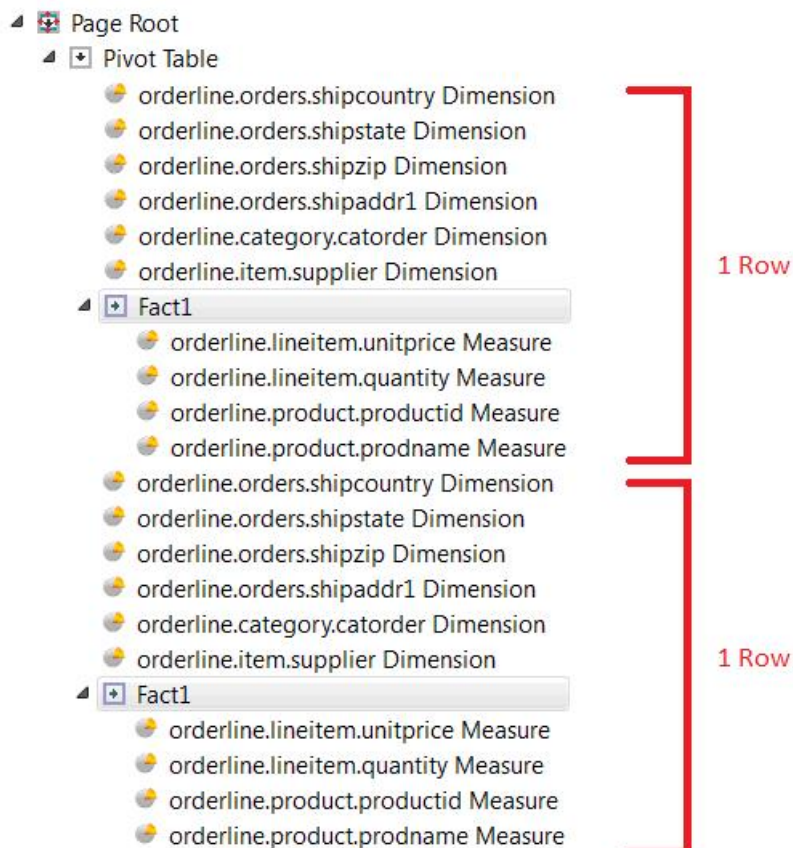


Figure 368: title

Expressions in properties

- [Overview](#)
- [Using the RTL Expression Language](#)
 - [Operators](#)
 - [Conditional Expressions](#)
 - [Operands](#)
 - [4GL Variables](#)
 - [Examples](#)
- [Using RTL Classes](#)
 - [Classes](#)
 - [Members](#)
 - [Examples](#)
- [Using the PXML Expression Language](#)
 - [Units of Measure](#)
 - [Variables](#)
 - [Functions](#)
- [Substituting 4GL Variables for Constants](#)

See also: [Dimension Resolver](#)

Overview

To define an object, values are specified for the object's properties.

The value for a [property](#) of a report item can be a literal value, or it can be derived from an expression that is written using the RTL Expressions language. RTL Expressions allow you to define runtime values for any property of a report item, except for those properties that display a specific set of valid values in a dropdown listbox.

Click the Value field to enter an expression in a field of the Properties View. To enter longer values, or obtain hints while typing the entry, click the **fx** button of the desired property to open the RTL Expression Editor:

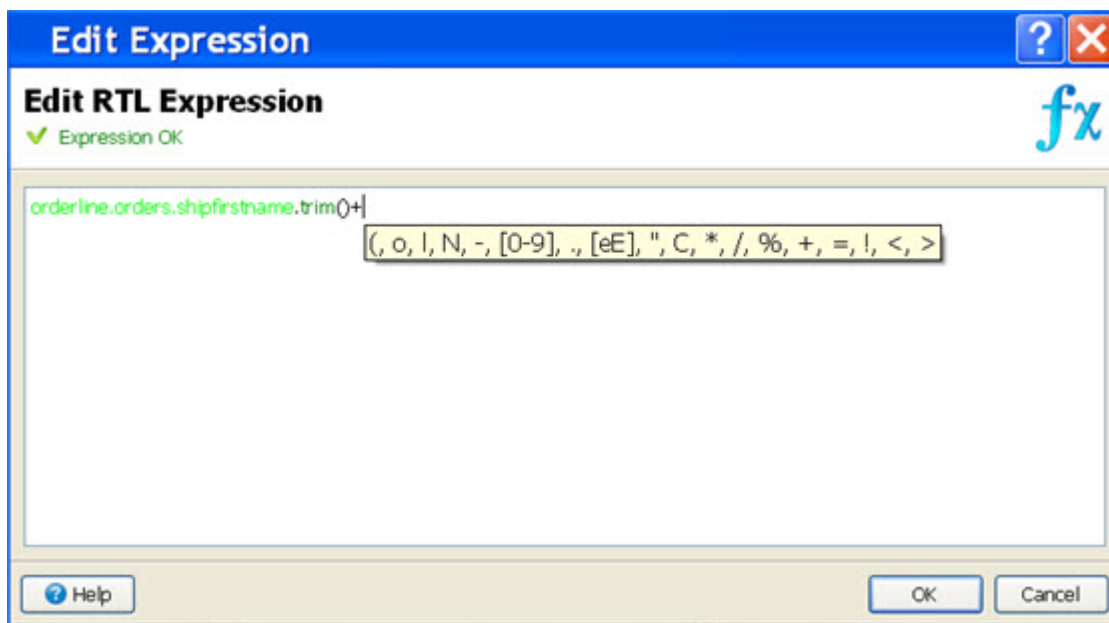


Figure 369: Edit Expression dialog

Press CTRL+SPACE and a Code Completion box appears, containing a list of the valid choices based on the context.

RTL Expressions

RTL Expressions:

- **are typed** - an expression is composed of items of different types and the expression returns a particular type. The [properties](#) in the Properties View are also typed. *Any expression entered as a value for a property must return the specified type.* For example, the [text](#) property of a [WordBox](#) has a type of String. So, any RTL expression for the text property is expected to yield a String. See [Report Element Properties](#) for a list of each report property and its type.
- **closely follow the Java™ syntax** for expressions. The main difference is that the "new" keyword is not supported; it is not possible to create and subclass objects. RTL Expressions loosely follow the Java™ evaluation semantics (operator precedence, evaluation order, and so on).
- **can use 4GL variables.** The variable is converted to a specific type within the expression (See [Conversion table](#)).

[RTL Classes](#) provide member functions (methods) and member variables that you can use in your expressions. There is a class for each type of a report property.

Important: Numeric data types are limited to 15 significant digits. To avoid problems with totals in reports, do not calculate the value of an item that is part of a total using an RTL expression. Perform any calculations in the BDL program.

Using the expression language

An Expression is a sequence of operands, operators, and parentheses that the runtime system can evaluate as a single value.

The RTL Expression language follows the Java™ syntax for expressions and evaluation semantics. Expressions can include these components:

- [Operators](#) on page 715
- [Conditional Expressions](#) on page 716
- [Operands](#) on page 716
- [4GL Variables](#) on page 716

Operators

Table 177: RTL Operators

Operator	Description	Example	Precedence
<code>%</code>	Arithmetic: Modulus	<code>x % 2</code>	8
<code>*</code>	Multiplication	<code>x * y</code>	7
<code>/</code>	Division	<code>x / y</code>	7
<code>+</code>	Addition	<code>x + y</code>	7
<code>-</code>	Subtraction	<code>x - y</code>	6
<code>+</code>	Concatenation	<code>string + string</code>	5
<code><</code>	Relational/Boolean: Less than	<code>numeric < 100</code>	4
<code><=</code>	Less then or equal to	<code>numeric <= 100</code>	4
<code>></code>	Greater than	<code>numeric > 100</code>	4
<code>>=</code>	Greater than or equal to	<code>numeric >= 100</code>	4
<code>==</code>	Equal to	<code>numeric == 100</code>	4
<code>!=</code>	Not equal to	<code>numeric <> 100</code>	4
<code>!</code>	Logical inverse (NOT)	<code>!(x = y)</code>	3
<code>&&</code>	Logical intersection (AND)	<code>expr1 && expr2</code>	2
<code> </code>	Logical union (OR)	<code>expr1 expr2</code>	1

The first column in the table describes the precedence order of the operators, listed highest to lowest. For example, the `%` modulus operator has a higher precedence than the `*` operator. Parentheses can be used to overwrite the precedence of operators.

Conditional Expressions

Conditional expressions allow you to express IF/ELSE statements.

Syntax:

```
Boolean-expression?expression-1:expression-2
```

The ? operator indicates that this expression is conditional; the return value is dependent on the result of the Boolean expression. If the Boolean expression is TRUE, the first expression is the return value; otherwise, the second expression is the return value.

You can use the `null` keyword in the ternary conditional operator. The “if then” and “if else” operands can be either expressions or the keyword `null`. A property whose RTL expression yields “null” is not set. This is useful in cases where a property should be set only when a certain condition is met. Consider the case where the background color of a WORDBOX should be set to red when a variable value `x` drops below a value of 10. The expression for this would be:

```
x<10?Color.RED:null
```

Operands

Operands include:

- Literal values
- Other expressions
- 4GL Variables
- [RTL Class Members](#)
 - Objects
 - Methods (returning a single value)

A literal value for a string in an expression should be delimited by double quotes: "Test".

4GL Variables

The data types of 4GL variables are taken into account when constructing expressions. For every 4GL variable an object is created that is either an instance of a `FGLNumericVariable` or an `FGLStringVariable`. These objects hold the value of the 4GL variable, and at the same time they contain a member variable **value** which also contains the value. For this reason, it is legal to write "order_line.itemprice" in your expression as a shortcut for "order_line.itemprice.value". Both types of objects have these specific member variables defined:

- **value**- value of the 4GL variable
- **caption**- the title of the field
- **name**- the name of the variable
- **type** - the RTL type of the variable
- **isoValue**- the locale and formatting-independent representation of the value of the variable

The conversion table lists 4GL data types and the type into which they are converted within an RTL expression:

Table 178: 4GL data types and the type into which they are converted within an RTL expression

4GL type	Corresponding RTL type
CHAR, VARCHAR, STRING and TEXT	FGLStringVariable
DATE, DATETIME and INTERVAL	FGLNumericVariable

4GL type	Corresponding RTL type
INTEGER, SMALLINT, FLOAT, SMALLFLOAT, DECIMAL and MONEY	FGLNumericVariable , limited to 15 significant digits. The value of a number larger than 15 digits will be truncated, and the resulting number is rounded. For example, 12345678901234567 will be rounded to 123456789012346.

Important: To avoid problems with inaccurate totals on a report due to rounding, do not perform RTL arithmetic on either the individual values or the total value; calculate the value of the item in the BDL program instead, before passing the value to the report.

Examples

For the purpose of these examples, `order_line` has been replaced with `order`.

1. To add 10% to the itemprice: `order.itemprice*1.10`

The data item `order_line.itemprice` is converted to a Numeric type, so we can use the Numeric operators. In order to display the result of a Numeric expression in a WordBox, we must convert the result to a String. See [Example 1](#) in the Using RTL Classes section.

2. Let's add 10% to the item price conditionally, depending on the value: `order.itemprice<100?order.itemprice*1.10:order.itemprice`

The condition in this Boolean expression tests whether the itemprice is greater than 100; if so, the value returned is 110% of the itemprice; otherwise, the value returned is simply the itemprice.

3. To set the font of a report item to italic when the 4GL variable **order_line.lineitemprice** exceeds \$20, we must create an expression for the **fontItalic** property: `order.lineitemprice>20`

The property **fontItalic** is of type boolean, so any RTL expression that we use for that property must return a boolean value (TRUE/FALSE). Any of the relational operators yields a boolean, so the type of the returned value of this expression is a boolean (The expression will return TRUE if the lineitemprice exceeds 20).

Note: A numeric value by itself is not a boolean value as it is in some programming languages.

Using RTL classes

RTL expressions do not contain the primitive data types "byte", "short", "int", "long", "float", "double", "boolean" and "char". Instead, everything is expressed as objects. All methods are member functions. There are no global functions.

Basic Object Classes

There are object classes for each type of the report item properties. See the [Properties](#) documentation to identify the type of a specific property.

The basic object class types for properties are:

- **String** - contains methods used for all string operations
- **Numeric** - contains methods used for all numeric operations; the class has the precision of a double and the arithmetic operators are defined for objects of this type.
- **Boolean** - contains methods used for all logical operations
- **Color** - contains methods and static member variables related to color.
- **Enum** - a set consisting of a class for each property of this type; each class contains static member variables that provide a list of valid values for the corresponding property.
- **Date Class** - a class representing a Date value; contains methods for parsing and formatting strings.

Members

Instance Member Methods

Instance Member methods are called on an object instance. You can get an object instance by referencing a 4GL variable or by calling a method on another object. You can also use a literal value as an object instance.

When you invoke the method, it is prefixed with the object instance name and the "." character.

Examples of instance methods are expressions like `order_line.customer_name.trim()`. This is valid because the 4GL variable `order_line.customer_name` has a CHAR data type, which is converted within the RTL Expression to an object of the type String. And, the method `trim()` is a member function of a String object.

Methods always yield objects, so it is also legal to call methods on the return value of a method.

Static Member Methods

Static Member methods do not require an object instance. When you invoke the method, it is prefixed with the classname and the "." character.

Static Member Objects

Static Member objects are member variables that do not require an object instance. The objects are prefixed with the classname and the "." character. Examples of static member objects are expressions like `Color.RED` or `Numeric.PI`.

Examples

1. This example concatenates the first and last names of a customer, using the trim method of the String class and the + operator:

```
order_line.shipfirstname.trim()+" "+order_line.shiplastname.trim()
```

This expression prints the first name (trimmed of trailing blanks), a string consisting of a single space, and the last name (trimmed). Use double quotes instead of single quotes to delimit strings.

2. Parenthesis can be used to change the order of operations. For example:

```
(order_line.unitprice+10).toString()
```

Parentheses are used to force the addition to be done prior to the conversion to a String.

3. This conditional expression used in the `color` property of the `order_line.unitprice` `WordBox` will change the color to red if the value is less than 20:

```
order_line.unitprice<20?Color.RED:Color.BLACK
```

This expression specifies that the return value when the boolean expression is TRUE is the static member variable RED of the Color class, otherwise the return value is the static member variable BLACK.

4. It is legal to call methods on the return value of a method. For example, this is a valid expression:

```
order_line.customer_name.trim().toUpperCase().substring(1)
```

Figure 370: order_line.customer_name.trim().toUpperCase().substring(1)

In this example, the object **order_line.customer_name** is a BDL CHAR variable; this variable is assigned to the String type. The String method **trim()** is called first, returning the String object a. The method **toUpperCase()** is called for the object a, returning object b which will be in upper case. Finally, the method **substring()** is called for the object b, returning object c. If the customer_name is "Springs", the resulting object c is the string "PRINGS".

There are many additional examples of expressions in the properties of report elements defined in the 4rp programs that are part of the GRWDemo project.

Using the PXML expression language

Genero Report Writer provides the PXML Expression language to define the value of a property that is of the PXML (dimension) type.

Tip: The type of each property is listed in the [Properties page](#) of the Report Writer documentation.

A PXML expression always yields a Numeric value. The value is expressed in units of measurement. It is legal, for example, for the value to be **10in**. If no unit is specified, the unit is presumed to be points. When you specify a value in units, it is converted internally to its equivalent value in points.

Units of Measure

The most commonly used units are:

- **point|pt**
- **pica|pc**
- **inch|in**
- **cm**
- **mm**

See [Dimension Resolver](#) for additional explanations and examples of the units that can be used.

Variables

These variables can be used in any PXML expression to define the layout dynamically:

- **max** - the maximum extent of the current parent box
- **min** - the minimum extent of the current parent box
- **rest** - the remainder of the current parent box

For example, to center an element in its parent container you can use the max variable for these properties:

Table 179: Centering an element

Property	Value
x	max/2
y	max/2
anchorx	0.5
anchory	0.5

You can use the **rest** variable in the **Y-Size** property of a **MiniPage** container to force a page break by consuming the remainder of the container:

Table 180: Using rest

Property	Value
Y-Size	rest

Functions

The PXML Expression language has a 4GL-like syntax. The most commonly used functions are:

- **max(valueA, valueB)** - this is a function, not the variable listed in [Variables](#) on page 719!
- **min(valueA, valueB)**
- **length(value)**
- **width(value)**

For example, this expression uses the functions **max** and **width** :

```
max(10cm,width("HELLO"))
```

In this example, the report engine first calculates the width of the string "HELLO", taking the current font metrics into account. It then determines which is larger (10cm or the calculated width of "HELLO") and returns the larger value.

Substituting 4GL variables for constants

You may want your expression to depend on a data variable rather than on the constant string, such as "HELLO".

For this, we use RTL expressions embedded in curly braces.

Note: We are now mixing two languages. The content within the braces is RTL, the content outside the braces is a PXML expression.

The rules for embedding RTL in PXML are:

- wherever a numeric constant is allowed in PXML, you can insert an RTL numeric expression (enclosed in curly braces) instead.
- wherever a string constant is allowed in PXML, you can insert a RTL string expression (enclosed in curly braces) instead.

For example, consider this expression:

```
max(10cm,width("HELLO"))
```

This PXML expression contains one numeric constant ("10") and one string constant ("HELLO"). These constants can be replaced by data variables, enclosed in curly braces:

```
max({order_line.titlewidth}cm,width({order_line.title}))
```

For this expression to be legal, the variable **order_line.titlewidth** has to be of type Numeric, and the variable **order_line.title** has to be of type String.

Note: You cannot construct a dynamic expression where the function names (such as `max` or `width`) or the unit names (such as `cm`) are dynamic.

Expression examples

Examples of common expressions used by report writers.

Check a field for a value

You want to check whether a field contains a value, is empty, or is null and act accordingly.

For this example, the data source includes the field `orderline.order.contact`, and the field is a `STRING`. The field either contains a value, is an empty string, or is null.

To test whether it contains a value, you write an expression:

```
orderline.order.contact.trim().length()>0
```

This expression will evaluate to either `TRUE` or `FALSE`.

To explicitly test for an empty string, there are two options:

```
orderline.order.contact.trim().length()==0
```

```
orderline.order.contact.isEmpty()==TRUE
```

To explicitly test for null:

```
orderline.order.contact.isNull()==TRUE
```

Use in the Visibility Condition property

Expressions that evaluate to `TRUE` or `FALSE` can be used in the **Visibility Condition** (`visibilityCondition`) property. If the expression evaluates as `TRUE`, then the instance of the element will appear in the report. If the expression evaluates as `FALSE`, the instance of the element (to include all its children, i.e. the entire element tree) is removed from the report. If you are using relative positioning, all sibling elements after this element in the report structure shift accordingly, reclaiming the space that the element would have occupied.

Use in the Text property

You can use these expressions when defining the **Text** (`text`) property. The `text` property specifies the text to be drawn.

```
orderline.orders.contact.trim().length()>0?orderline.orders.contact:" "
```

In this expression, the expression evaluates to `TRUE` when the length of the trimmed field is greater than zero and the field value is printed (to include any leading or trailing spaces). If the length is not greater than zero, the field is identified as not having a value and an empty string is printed; the vertical allocated space for that field remains in the report.

An alternate expression could simply be:

```
orderline.orders.contact.trim()
```

Tip: When you use a character type with a fixed length for a field (such as `CHAR(N)`), you typically need to add `.trim()` or `.trimRight()` to remove trailing spaces. You can avoid this by using the `STRING` data type. With the `STRING` data type, the value is not padded with trailing spaces unless trailing spaces are explicitly set.

```
DEFINE field1 CHAR(5),
DEFINE field2, field3 STRING
LET field1="ABC"    -- you end up with "ABC  "
LET field2="ABC"    -- you end up with "ABC"
```

```
LET field3="ABC " -- you end up with "ABC ", as explicitly specified
```

Report Designer Reference

- [Report Design Elements \(The Toolbox\)](#) on page 722
- [Report Element Properties](#) on page 736
- [Bar Codes](#) on page 764
- [RTL Classes Overview](#) on page 801
- [Dimension Resolver](#) on page 801

Report Design Elements (The Toolbox)

The Toolbox contains report object that can be placed on a report design document.

- Simple Containers
 - [Horizontal Box \(Layout Node\)](#)
 - [Vertical Box \(Layout Node\)](#)
- Propagating Containers
 - [Horizontal Box \(Mini Page\)](#)
 - [Vertical Box \(Mini Page\)](#)
 - [Page Root \(MiniPage\)](#)
 - [Stripe \(MiniPage\)](#)
- Drawables
 - [Word Box](#)
 - [Word Wrap Box](#)
 - [HTML Box](#) on page 725
 - [Decimal Format Box](#) on page 725
 - [Page Number \(PageNoBox\)](#) on page 727
 - [Image Box](#) on page 727
 - [Table](#) on page 729
- Business Graphs
 - [Map Chart](#) on page 731
 - [Map Chart Item](#) on page 731
 - [Category Chart](#) on page 732
 - [Category Chart Item](#) on page 732
 - [XY Chart](#) on page 732
 - [XY Chart Item](#) on page 732
 - [Pivot Table](#) on page 733
 - [Pivot Table Hierarchy Value](#) on page 733
 - [Pivot Table Fact](#) on page 734
 - [Pivot Table Measure](#) on page 734
- References
 - [Reference Box](#) on page 735
 - [Info Node](#) on page 735
- Bar Codes
 - [Bar Code Box](#) on page 735

Simple Containers

The Simple Containers section of the toolbox contains those containers that do not propagate. These containers are used to group and organize objects on a page.

Horizontal Box, Vertical Box (Layout Node)

Container: A Layout Node is a rectangular area in the Report Design page. A Layout Node does not propagate; the content is not allowed to overflow the container. As a result, a Layout Node can be used for content that should be kept together on the page.

The **Horizontal Box** is a Layout Node with the Layout Direction property set **lefttoright**; elements in the box are laid out across the page. The **Vertical Box** is a Layout Node with the layout direction property set **toptobottom**; elements in the box are laid out down the page.

Properties

Select the object on the Report Design page to display its properties in the Properties View.

You can change the object's default appearance by setting the values of these properties:

[name](#), [color](#), [bgColor](#), [fontName](#), [fontSize](#), [X-SizeAdjustment](#), [Y-SizeAdjustment](#), [fontBold](#), [fontItalic](#)

Some specific properties allow you to [define borders, margins, and padding](#) for the boxes:

[marginWidth](#), [marginRightWidth](#), [marginBottomWidth](#), [marginLeftWidth](#), [marginTopWidth](#), [borderWidth](#), [borderRightWidth](#), [borderBottomWidth](#), [borderLeftWidth](#), [borderTopWidth](#), [paddingWidth](#), [paddingRightWidth](#), [paddingBottomWidth](#), [paddingLeftWidth](#), [paddingTopWidth](#), [borderStyle](#), [borderRightStyle](#), [borderBottomStyle](#), [borderLeftStyle](#), [borderTopStyle](#), [borderColor](#), [borderRightColor](#), [borderBottomColor](#), [borderLeftColor](#), [borderTopColor](#), [roundedCorners](#)

The [x-Size](#) and [y-Size](#) properties specify the INNER size of the box. Adding borders, for example, will increase the overall size.

Layout Nodes can be used as a container for content that must print at a specific part of the MiniPage, a page header, for example, by setting its [section](#) property.

The [clip](#) property can be used to clip the object box and its content along the sides. This property applies to all layout nodes.

Other properties have values that are derived from the type and position on the page, adjust automatically if the object is moved or re-sized, and need not be changed manually:

[type](#), [baselineType](#), [layoutDirection](#), [swapX](#), [alignment](#), [scaleX](#), [scaleY](#), [x](#), [y](#), [anchorX](#), [anchorY](#), [floatingBehavior](#)

Propagating Containers (Mini Pages)

The Propagating Containers section of the toolbox contains those containers that propagate; if the container fills, a copy is generated and the extra content overflows to the copy.

Mini Page

Container: Mini Page. This container formats the report page into lines and columns.

A Mini Page is a propagating box. The boxes can handle unknown amounts of material; if the box is full, a copy is made and the leftover material flows to the copy or copies, as needed. A MiniPage cannot be used as a container for a [page header](#), [page footer](#), or separator.

Page Root (Mini Page)

Container: Mini Page. Page Root is the recommended base container when you start creating a report. It is a Mini Page with the height and width properties set to maximum. The container propagates; if it is full, a copy is generated and the extra content overflows to the copy.

Stripe (MiniPage)

Container: Mini Page. This container has the y-size property set to "max". This container is recommended for content that stretches horizontally across the report page (lines in a report, for example.) The container propagates; if it is full, a copy is generated and the extra content overflows to the copy.

Horizontal Box, Vertical Box (Mini Page)

Container: Mini Page. These containers have their orientation ([layoutDirection swapX \(Swap X\)](#) on page 756property) set to display the box content horizontally (lefttoright) or vertically (toptobottom). The containers propagate; if a container is full, a copy is generated and the extra content overflows to the copy.

Properties

These properties are specific to Mini Page:

[Hide PageHeader OnLastPage](#), [Hide PageFooter OnLastPage](#)

Additional properties are inherited from Propagating Box and [Layout Node](#). The property [floatingBehavior](#) allows you to specify whether the parent container will resize itself so that this Mini Page object is enclosed in the parent.

Drawables

The Drawables section of the toolbox contains a variety of objects that can hold static or dynamic values, such as text, numbers, HTML snippets, page numbers, images, and tables.

Word Box and Word Wrap Box

The Word Box and Word Wrap Box are layout objects for the display of text.

Word Box

Word Box (WordBox type) is a layout container, found in the Drawable group in the Tool Box view.

Use this object for a specified chunk of text, which will use the current font.

These properties are specific to Word Box:

[trimText](#), [underline](#), [strikethrough](#), [fidelity](#), [localizeText](#)

Word Wrap Box

Word Wrap Box (WordWrapBox type) is a layout container, found in the Drawable group in the Tool Box view.

This object is like a WordBox with paragraphs of uniform text.

These properties are specific to Word Wrap Box:

[trimText](#), [indent](#), [fidelity](#), [localizeText](#)

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

The [text](#) property specifies the string to be displayed in the WordBox or Word Wrap Box.

You can set the [textAlignment property](#) for a Word Box or Word Wrap Box to **left**, **right**, or **center**. The alignment does not influence page break positions even if the [indent](#) property is set to some value. For Word Wrap Boxes, the textAlignment property can also be set to **justified**.

The [localizeText](#) property enables the localization of text content in Word Boxes and Word Wrap Boxes.

Additional properties are inherited from [Layout Node](#).

Tip:

- Don't set the [Y-Size](#) (height) property on a Word Wrap box, because the element should typically grow in accordance with its content. If you set a fixed [Y-Size](#), you'll prevent that automatic enlargement.
- The text value of these boxes can be edited directly in the report design document; double-click the object and the input cursor will be placed in the text. The layout will be updated on each keystroke.
- To force a line break, use the newline character "\n".
- Beginning in version 2.4x, a Word Wrap Box can span pages; when the available vertical space for WORDWRAPBOX is not sufficient to display the entire text, the box now propagates the exceeding content to additional boxes, with the same behavior as a propagating [MINIPAGE](#).

Decimal Format Box

Use a Decimal Format Box for decimal numbers.

Decimal Format Box (DecimalFormatBox type) is a layout container, found in the Drawable group in the Tool Box view.

Use a Decimal Format Box for decimal numbers. It has a variety of features designed to make it possible to parse and format numbers in any locale, including support for Western, Arabic, and Indic digits. It also supports different kinds of numbers, including integers (123), fixed-point numbers (123.4), scientific notation (1.23E4), percentages (12%), and currency amounts (\$123). All of these can be localized. The value of the number is limited to 15 significant digits.

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

format

The value of the text property can also be edited directly in the report design document; double-click the object and the input cursor will be placed in the text. The layout will be updated on each keystroke.

HTML Box

An HTML Box displays an image of an HTML document in the report.

HTML Box (HTMLBox type) is a layout container, found in the Drawable group in the Tool Box view.

Use the [Location](#) property to specify the file name and path of the document whose image is to be displayed. Press the ... button to open a dialog and select the html file.

Note: The content of an HTML box cannot span pages.

Embedding HTML

To embed an image, we use a URL type that allows encoding the data in the body of the URL text. Such a URL starts with the protocol name "data" and a short description of the data, followed by a colon and the encoded data itself. The full syntax of data URLs is:

```
data: [<MIME-type>] [ ; charset=<encoding> ] [ ; base64 ] , <data>
```

See [data URI scheme](#) for a complete description of the concept and the syntax.

For our purposes, it is sufficient to support a simplified subset that omits the "charset" and assumes that characters are encoded in utf8. Image data is always encoded in [base64](#) while other data such as HTML content is typically "[Percent encoded](#)".

For HTML content a URL would have the form "data:text/html, <data>". To embed HTML, we use the data protocol syntax in the [Location](#) property of the HTMLBOX element. The syntax is:

```
data:text/html, followed by the percent encoded data of the html document
```

To automatically construct this URL for HTML documents, press the **...** button for the [Location](#) property. When the Open dialog displays, select the HTML file located in your file structure, and check the **Embed in document** checkbox at the bottom of the dialog:

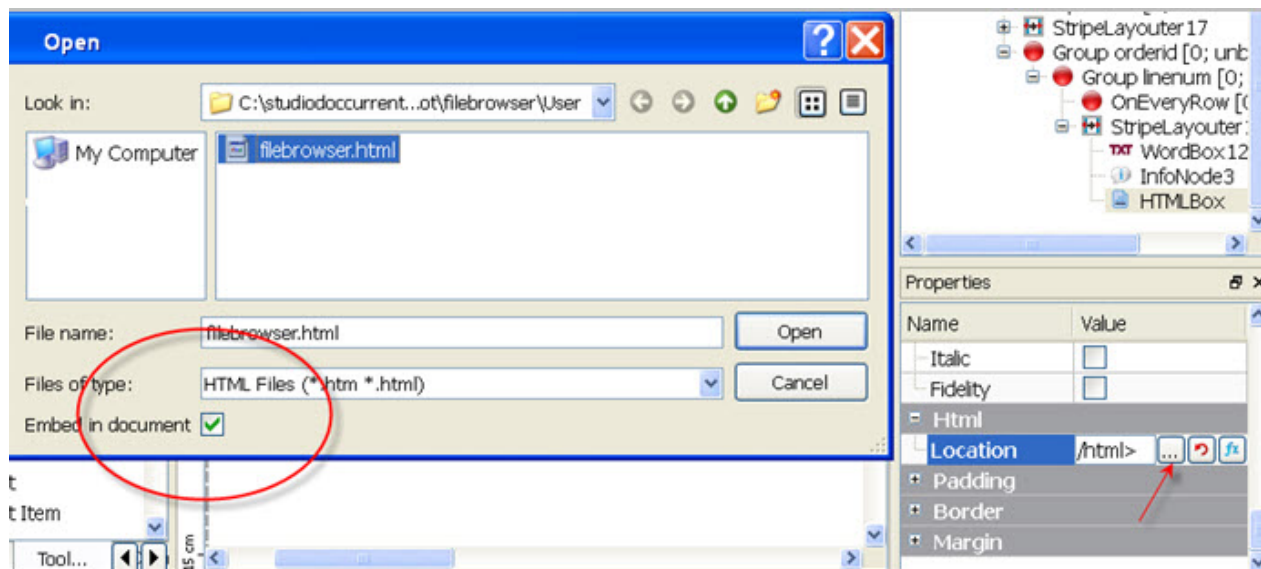


Figure 371: Embed in document checkbox

Populating HTML content from text variables

Data in text variables was typically input into a database via a TextEdit form field, with the textFormat style attribute set to **html**. See the Presentation Styles topic in the *Genero Business Development Language User Guide* for more details.

Press the **formula** button for the [Location](#) property of the HTML Box, and enter the expression, including the name of the text variable.



Figure 372: RTL Expression

The function `String.urlEncode()` is used to encode the data using percent encoding.

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

The [Location](#) property is used to specify the html content.

Page Number (PageNoBox)

Use this object for page numbers.

Page Number (PageNoBox type) is a layout container, found in the Drawable group in the Tool Box view.

In order to provide for virtual pages (multiple logical pages on one physical page) a [pageName](#) property can be specified to identify the logical page.

Specific functions are available to allow you to calculate an expression for the page number such as [Page N of M](#), using the property [textExpression](#). If a value is provided for [textExpression](#), the [pageName](#), the [pageNoOffset](#), and [pageNoFormat](#) properties are ignored.

If values for either the [textExpression](#) property or the [text](#) property are not set, a default length is calculated. See [Calculating the page number string](#).

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

These properties are specific to PageNoBox:

[pageName](#), [pageNoOffset](#), [pageNoFormat](#), [textAlignment](#), [textExpression](#)

Additional properties are inherited from [Word Box](#).

Image Box

Use this object for images.

Image Box (ImageBox type) is a layout container, found in the Drawable group in the Tool Box view.

Drawable: Use this object for images. On all platforms GIF, JPEG, PNG, BMP, WBMP, and SVG formats are supported. The images are cached on a per document basis. The default resolution is the current screen resolution.

Use the [Location](#) property to specify the location of the image to be rendered. Location values are URLs supporting the protocols "http", "file" and "data". The URL can be:

- absolute - Press the ... button to open a dialog and select the image located in your file structure; this populates the file name and path of the image to be rendered.
- variable - using an RTL expression to provide the image name as a variable. For example: `./images/database/"+orderline.product.prodpic.trim()`

If you want to preserve the aspect ratio of an image, set the value of either length (y-size) or width (x-size), allowing Report Writer to calculate the corresponding value. If you set both properties, the resulting image will appear distorted.

SVG images - Using SVG images instead of bitmap images can substantially reduce the document size, which is particularly useful when PDF documents are produced. When providing the SVG content from a 4GL variable, use the mime type "image/svg" so that the url looks something like "data:image/svg,..." when read from a string variable and "data:image/svg;base64,..." when read from a BLOB. The currently supported SVG version is "1.2 Tiny" (See <http://www.w3.org/TR/SVGTiny12>).

Embedding images

To embed an image, we use a URL type that allows encoding the data in the body of the URL text. Such a URL starts with the protocol name "data" and a short description of the data, followed by a colon and the encoded data itself. The full syntax of data URLs is:

```
data:[<MIME-type>][;charset=<encoding>][;base64],<data>
```

See [data URI scheme](#) for a complete description of the concept and the syntax.

For our purposes, it is sufficient to support a simplified subset that omits the “charset” and assumes that characters are encoded in utf8. Image data is always encoded in [base64](#) while other data such as HTML content is typically “[Percent encoded](#)”.

For images the simplified URL has the form "data:/text:xxx;base64,<data>" where xxx is one of “png”, “jpg”, “gif” or “bmp”. For an image, the syntax is:

```
data:image/png;base64, followed by the base 64 encoded bitmap data of the image
```

To automatically construct this URL for an image, press the ... **button** for the [Location](#) property. When the Open dialog displays, select the image file located in your file structure, and check the **Embed in document** box at the bottom of the dialog:

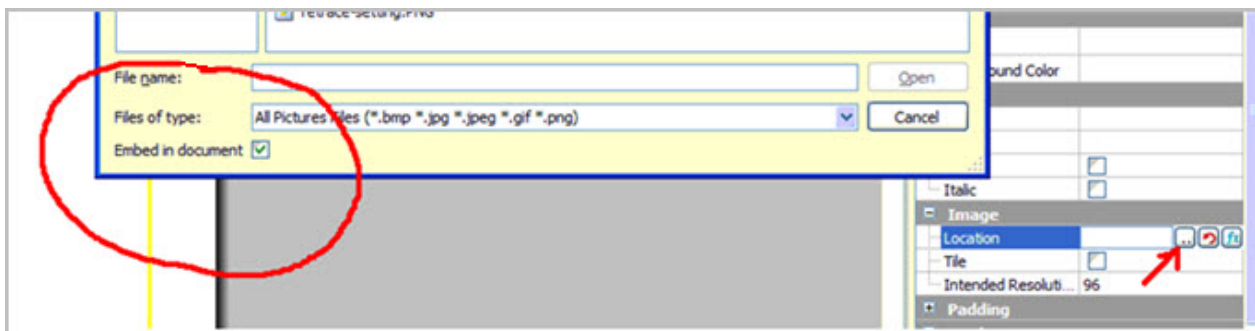


Figure 373: Embed in document checkbox

Populating images from blob variables

If you drag a BYTE variable from the Data View onto the Report Design, an Image Box object will be created. If the variable was named "imageblob" then the [Location](#) property would automatically be filled with this formula:

```
data:image/jpg;base64,"+imageblob
```

where **imageblob** is the name of the blob

This has the same effect as pressing the **formula button** for the [Location](#) property, and entering the formula including the blob variable name.

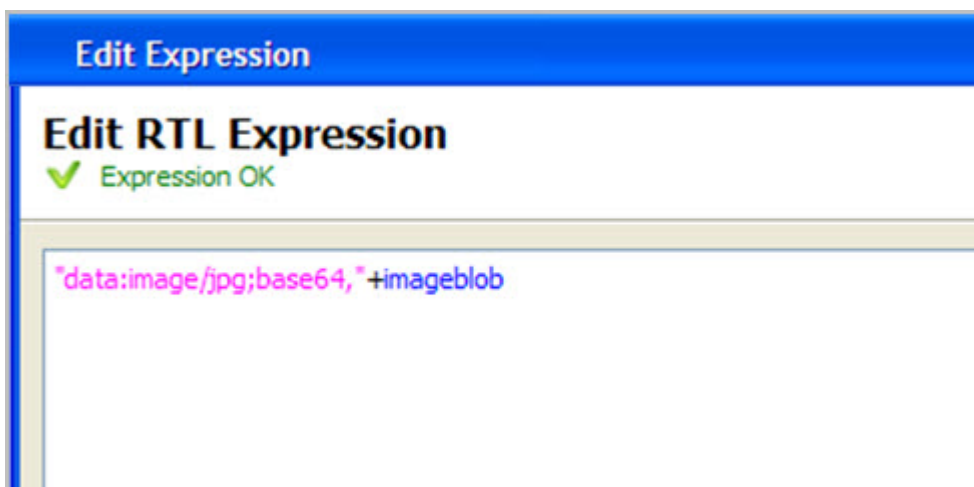


Figure 374: Entering blob variable name in RTL Expression editor

Note: Since a blob variable may contain images of various types, the implementation ignores the image type declared in the formula, and looks at the encoded data itself to determine the image type. This formula would work for blobs that were of **png** type also.

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

The [Location](#) property is used to specify the image.

These properties are specific to Image Box:

[Intended Resolution](#), [Location](#), [Fill](#)

Note: Beginning with version 2.4x, the **tile** property is replaced by a new property, **fill**. If you open a document containing the tile property, it will be automatically replaced by **fill**.

Fallback Image

A *fallback image* is the image to be displayed if the requested image is not found. To specify a fallback image, set the `GRE_DEFAULT_IMAGE_URL` environment variable to the image URL. The image URL can be a relative URL; it resolves relative to the location of the form design (`4rp`) file.

Table

A table object has the ability to display data in columns and rows.

Table (Table type) is a layout container, found in the Drawable group in the Tool Box view.

When you drag a table onto a report, it creates a table with a default of two rows and three columns. Of the two rows, one is a header row (Any Page Header) and one is a body row (Body). You can add and remove rows and columns, size the table or its components, merge columns, define its borders and padding and much more.

Table Structure

In the Report Structure, you can view the table structure.

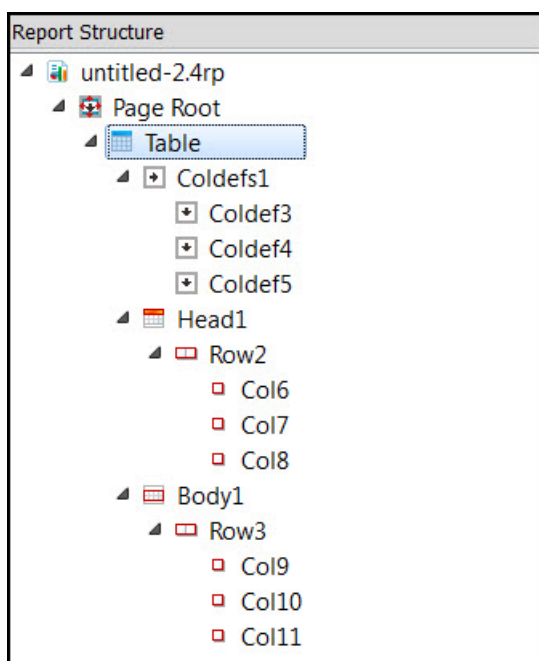


Figure 375: Table Object as viewed in the Report Structure

The top-level element is the Table element. It contains three child elements, which make up the parts of the table. These elements are the column definitions, the head, and the body.

The *column definitions* (or *coldefs*) define the basic properties for the columns in this table object. These properties include settings for padding, width, and alignment.

The head section contains the heading rows.

The body section contains the body rows.

Table Properties

Properties specific to the table involve rules, borders, and padding.

A *rule* refers to a line that separates two rows or two columns. Rule-related properties include [Rule](#), [Rule Color](#), [Horizontal Rule](#), and [Vertical Rule](#).

The *border* refers to the border around the table. Border-related properties include [Border](#), [Border Color](#), [Top Border](#), [Left Border](#), [Bottom Border](#), and [Right Border](#).

Padding refers to the space between a cell boundary and the value contained within. Padding-related properties include [Padding](#), [Horizontal Padding](#), and [Vertical Padding](#).

Column Properties

Column properties are specific to the column selected. Any column property set overrides the same property set for the table.

You can set the padding for the cells of a column. Padding-related properties include [Padding](#), [Horizontal Padding](#), and [Vertical Padding](#).

You can set the width of a column using [Proportional Width](#) or [Fix Width](#).

You can set the alignment of a value within the cells of a column with the [Horizontal Alignment](#) and Vertical Alignment properties.

Cell Properties

Cell properties are specific to the cell selected. Any cell property set overrides the same property set for the table or the column.

You can set the padding for the cells of a column. Padding-related properties include [Padding](#), [Horizontal Padding](#), and [Vertical Padding](#).

You can set the alignment of a value within the cells of a column with the [Horizontal Alignment](#) and [Vertical Alignment](#) properties.

You can merge cells by setting the [Column Span](#) property.

Demos

The **Reports** demo project includes two report design documents showing reports that include a table object.

- The `TableDemo.4rp` shows a report design document with a simple table containing five columns and two rows. One row is the **Any Page Header** row, while the other row is the **Body** row.
- The `GroupedTableDemo.4rp` shows a report design document where the table is more complex, with several header and body rows. For each header row, the section property specifies whether it is the First Page Header, Any Page Header, and so on. Each body row is created with a purpose - to show a row of data, to show the sum of a group of rows, and so on. Some of the cells in the summary rows span columns. Triggers are used to determine when each of the body rows is output to the table in the report.

Business Graphs

The Business Graphs section of the toolbox contains a variety of chart objects (map charts, category charts, XY charts) and pivot table objects.

Map Chart

The MAPCHART element defines the header for an abstract map dataset that can be used for creating a variety of one dimensional graphs such as pie charts.

Map Chart (Mapchart type) is a layout container, found in the Business Graphs group in the Tool Box view.

The MAPCHART element defines the header for an abstract map dataset that can be used for creating a variety of one dimensional graphs such as pie charts. The map items are defined using the ITEM element. The resulting chart is drawn automatically. See [Working with Business Graphs](#) for additional information.

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

[title](#), [valuesTitle](#), [keysTitle](#), [drawAs](#), [fidelity](#)

The valid values of **drawAs** for this object are: Pie|Pie3D|Ring|Bar|Bar3D|Table|SortedTable|AggregatedTable. The default is a Pie.

The **fidelity** property applies only if the chart is drawn as a table (`drawAs="Table"`).

Map Chart Item

A Map Chart Item defines the data value items for a Map Chart.

Map Chart Item is found in the Business Graphs group in the Tool Box view.

A Map Chart Item defines the data value items for a [Map Chart](#).

Properties

Select the object on the Report Design page to display its properties in the Properties View.

[key](#), [value](#)

Category Chart

A Category Chart defines the header for an abstract category dataset that can be used for creating a variety of two dimensional charts.

Category Chart (Categorychart type) is found in the Business Graphs group in the Tool Box view.

The CATEGORYCHART element defines the header for an abstract category dataset that can be used for creating a variety of two dimensional charts such as category charts. The categories are defined by the CATEGORY element and its "key" attribute, which has to be unique within a CATEGORYCHART. Within a CATEGORY, CATEGORYITEMS define the values within the category. Within one category, the "key" values of individual CATEGORYITEM elements has to be unique. The resulting chart is drawn automatically. See [Working with Business Graphs](#) for additional information.

Properties

Select the object on the Report Design page to display to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

[title](#), [valuesTitle](#), [keysTitle](#), [drawAs](#), [fidelity](#)

The valid values of **drawAs** for this object are:

Bar|Bar3D|Area|StackedBar|StackedArea|Line|

Line3D|Waterfall|Table|SortedTable|AggregatedTable. The default is a Bar.

The **fidelity** property applies only if the chart is drawn as a table (drawAs="Table").

Category Chart Item

A Category Chart Item defines the data value items for a Category chart.

Category Chart Item is found in the Business Graphs group in the Tool Box view.

A Category Chart Item defines the data value items for a [Category chart](#).

Properties

Select the object on the Report Design page to display to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

[category Key](#), [key](#), [value](#)

XY Chart

An XY Chart defines the header for an abstract XY dataset that can be used for creating a variety of XY-Plots such as line or scatter plots.

XY Chart (XyChart type) is found in the Business Graphs group in the Tool Box view.

The XYCHART element defines the header for an abstract XY dataset that can be used for creating a variety of XY-Plots such as line or scatter plots. The XY data is defined by XYITEM elements. The resulting plot is drawn automatically. See [Working with Business Graphs](#) for additional information.

Properties

[title](#), [xAxisTitle](#), [yAxisTitle](#), [drawAs](#), [fidelity](#)

Value values of **drawAs** for this object are: Polar|Scatter|Area|Line|Step|StepArea|TimeSeries|Table|SortedTable. The default is Line.

The **fidelity** property applies only if the chart is drawn as a table (drawAs="Table").

XY Chart Item

An XY Chart Item defines the data value items for an XY Chart.

XY Chart Item is found in the Business Graphs group in the Tool Box view.

It defines the data value items for an [XY Chart](#).

Properties

Select the object on the Report Design page to display to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

[seriesTitle](#), [x](#), [y](#)

Pivot Table

The PIVOTTABLE element is the enclosing element of an abstract pivot dataset that can be used for creating a variety of multidimensional outputs such as tables and charts.

Pivot Table is found in the Business Graphs group in the Tool Box view.

The resulting data is drawn into a box defined by [X-Size](#) and [Y-Size](#). If these are not defined, the view will expand to whatever space it can claim in the parent.

The PIVOTTABLE element is part of the Business Graphs group in the Report Designer Tool Box.

Depending on the visualization type (specified by the [drawAs](#) property), the output may span several pages. For these visualization types (such as Tables), the enclosing containers should support propagation.

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

In addition to the attributes available for LAYOUTNODE, the PIVOTTABLE element has the following properties:

- [title](#)
- [drawAs](#)

The **drawAs** property specifies the type of output that is rendered from the data. Depending on the type selected and the number of available dimensions, the rendering is delegated to the map chart, category chart, XY chart or table element. In case that the number of selected dimensions outnumbers the respective number in the selected visualization, the exceeding dimensions and measures are ignored. The values are assigned from left to right so that if for example a pivot table with 4 dimensions and 3 measures is drawn as a category chart which has only 2 dimensions and one measure, then the chart will be drawn using the first two dimensions and the first measure from the pivot table's columns. Selecting "Table" causes the output to be drawn in tabular form, displaying all selected columns of the pivot table.

The valid values of **drawAs** for this object are: Area | Bar | Bar3D | Line | Line3D | Pie | Pie3D | Polar | Ring | Scatter | StackedArea | StackedBar | Step | StepArea | Table | TimeSeries | Waterfall | XYArea | XYStackedArea | XYLine . The default is a Table.

- [fidelity](#)

The **fidelity** property applies only if the chart is drawn as a table ([drawAs](#)="Table").

- [computeAggregateInnermostDimension](#)
- [hierarchiesInputOrder](#)
- [displaySelection](#)
- [displayRecurringDimensions](#)

Pivot Table Hierarchy Value

The HIERARCHY elements represent dimensions. An element represents both the declarative aspects of the column as well as the data value which is typically defined as an RTL expression.

Pivot Table Hierarchy Value is found in the Business Graphs group in the Tool Box view.

HIERARCHY elements are child elements of PIVOTTABLE and need to be located before a FACT element containing the measures of the row.

The HIERARCHY element is part of the Business Graphs group in the Report Designer Tool Box.

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

- [dataType](#) (default String)
- [title](#) (default is the empty string)
- [format](#) (default ---,---,---,--&.&&)
- [value](#)
- [enumValues](#)
- [computeTotal](#)
- [computeCount](#)
- [computeDistinctCount](#)
- [computeAverage](#)
- [computeMinimum](#)
- [computeMaximum](#)

Pivot Table Fact

Together with the HIERARCHY elements that can precede it the FACT element defines a table row.

Pivot Table Fact is found in the Business Graphs group in the Tool Box view.

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

- [outputOrder](#)
- [displaySelection](#)
- [displayFactRows](#) (default **true**)
- [topN](#)

Pivot Table Measure

A MEASURE element represents both the declarative aspects of the column as well as the data value.

Pivot Table Measure is found in the Business Graphs group in the Tool Box view.

The data value is typically defined as an RTL expression. MEASURE elements are child elements of FACT elements.

The MEASURE element is part of the Business Graphs group in the Report Designer Tool Box.

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

- [dataType](#) (default String)
- [title](#) (default is the empty string)
- [format](#) (default ---,---,---,--&.&&)
- [value](#)

References

The References section of the toolbox provides two objects: a Reference Box and an Info Node. These two objects typically work together in a report design.

Reference Box

A Reference Box allows you to create layout-dependent text output like "Total from previous pages: num".

Reference Box (ReferenceBox type) is a layout container, found in the References group in the Tool Box view.

This object allows you to create layout-dependent text output like "Total from previous pages: num".

Since the space to be allocated may not be known until the report is run, make sure that there is enough space available to display any possible text. Use the **text** property to provide an example, based on the underlying data type of the InfoNode object. This is only used to determine the maximum space to set aside. For example:

- Data types that are numeric - "000,000,000.00"
- Data types that are strings, for example CHAR(8) - "MMMMMMMM"

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

This object works in conjunction with an [Info Node](#) object, and its specific properties reference the Info Node:

- [InfoNode Name](#) - the name of the InfoNode to be referenced
- [default](#) - text to be displayed if the reference cannot be resolved. The default string is "-".

Additional properties are inherited from [PageNo Box](#), [Word Box](#).

See [DesignHowTo](#) for an example, and the demo programs provided with the product.

Info Node

The Info Node helps resolve some layout-dependent problems by enabling the use of references.

Info Node (InfoNode type) is a layout container, found in the References group in the Tool Box view.

This object helps resolve some layout-dependent problems by enabling the use of references. This node is invisible and does not consume space in the layout. This is the object referenced by a [Reference Box](#), to print layout specific information, such as a total from a previous page, for example.

Properties

The value is stored for querying by a Reference Box. The value is of type [String](#). If the data type of the field being referenced does not correspond to a String, the value must be converted.

See [DesignHowTo](#) for an example of this use, and the demo programs provided with the product.

Bar Codes

The Bar Codes section of the toolbox contains barcode objects

Bar Code Box

The Bar Code Box displays bar codes.

The Bar Code Box (BarCodeBox type) is a layout container, found in the Bar Codes group in the Tool Box view.

Use this object for bar codes. Example:



Figure 376: Sample Barcode

The currently supported types are listed in the topic [Bar Code type listing](#) on page 765. For licensing reasons, it may be necessary for the user to supply the fonts required to draw the text for some types of bar code.

Bar codes are drawn in nominal sizes. By setting the [scaleX](#) and [scaleY](#) properties it is possible to draw larger or smaller versions. It is further possible to force a particular width and/or length but specifying the desired extend value.

Specific functions are available to allow you to calculate an expression for the page number such as [Page N of M](#), using the property [codeValueExpression](#).

Properties

Select the object on the Report Design page to display its properties in the Properties View. You can change the object's default appearance by setting the values of its properties.

These properties are specific to Barcode Box:

[codeType](#), [fidelity](#), [noText](#), [codeValue](#), [check](#), [noDigits](#), [noCheckDigits](#), [thinToThickRelation](#), [thinToGapRelation](#), [controlCharacters](#), [codeValueExpression](#)

Additional properties are inherited from [Layout Node](#), and specific bar codes may have unique properties.

See [Bar Codes](#) for a description of all the possible bar code types.

Report Element Properties

Each element has associated properties.

Values for these properties may be literal, or they may be RTL expressions. If expressions are used, the resulting value must be of the specified data type of the property.

See [Using Expressions in Properties](#).

- [General Properties](#) on page 736
- [Properties related to margins and borders](#) on page 761
- [Properties for Report Document Metadata](#) on page 764

General Properties

General properties of a report element.

- [alignment \(Alignment\)](#) on page 739
- [anchorX \(Anchor X\)](#) on page 739
- [anchorY \(Anchor Y\)](#) on page 739
- [baselineType \(Baseline Type\)](#) on page 739
- [bBorder \(Bottom Border\)](#) on page 740
- [bgColor \(Background Color\)](#) on page 740
- [border \(Border\)](#) on page 740
- [categoryKey \(Category Key\)](#) on page 740

- [categoryTitle \(Categories Title\)](#) on page 740
- [check \(Check\)](#) on page 740
- [class \(Class\)](#) on page 740
- [clip \(Clip\)](#) on page 741
- [codeType \(Code Type\)](#) on page 741
- [codeValue \(Code Value\)](#) on page 741
- [codeValueExpression \(Code Value Expression\)](#) on page 741
- [color \(Color\)](#) on page 742
- [colspan \(Column Span\)](#) on page 742
- [computeAggregatesInnermostDimension \(Compute aggregates on the innermost dimension\)](#) on page 742
- [computeAverage \(Compute Average\)](#) on page 742
- [computeCount \(Compute Count\)](#) on page 742
- [computeDistinctCount \(Compute Distinct Count\)](#) on page 742
- [computeMaximum \(Compute Maximum\)](#) on page 743
- [computeMinimum \(Compute Minimum\)](#) on page 743
- [computeTotal \(Compute Totals\)](#) on page 743
- [controlCharacters \(Control Characters\)](#) on page 743
- [dataSymbolsPerLine \(Data Symbols per Line\)](#) on page 743
- [displayFactRows \(Display Fact Rows\)](#) on page 744
- [displayRecurringDimensions \(Display Recurring Dimension Values\)](#) on page 744
- [displaySelection \(Display Selection\)](#) on page 744
- [drawAs \(Draw As\)](#) on page 744
- [drawLabels \(Draw Labels\)](#) on page 745
- [drawLegend \(Draw Legend\)](#) on page 745
- [encoding \(Encoding\)](#) on page 745
- [enumValues \(Enum Values\)](#) on page 745
- [errorCorrectionDegree \(Error Correction Degree\)](#) on page 746
- [fidelity \(Text Fidelity\)](#) on page 746
- [fill \(Fill\)](#) on page 746
- [floatingBehavior \(Floating Behavior\)](#) on page 746
- [fontBold \(Bold\)](#) on page 746
- [fontItalic \(Italic\)](#) on page 747
- [fontName \(Name\)](#) on page 747
- [fontSize \(Size\)](#) on page 747
- [format \(Format\)](#) on page 747
- [fWidth \(Fix Width\)](#) on page 747
- [hAlign \(Horizontal Alignment\)](#) on page 747
- [hidePageHeaderOnLastPage \(Hide Page Header On Last Page\)](#) on page 748
- [hidePageFooterOnLastPage \(Hide Page Footer On Last Page\)](#) on page 748
- [hierarchiesInputOrder \(Hierarchies input order\)](#) on page 748
- [href \(href\)](#) on page 748
- [hPadding \(Horizontal Padding\)](#) on page 748
- [hRule \(Horizontal Rule\)](#) on page 749
- [id \(id\)](#) on page 749
- [indent \(Indent\)](#) on page 749
- [intendedResolution \(Intended Resolution\)](#) on page 749
- [key \(Key\)](#) on page 749
- [keysTitle \(Keys Title\)](#) on page 749
- [layoutDirection \(Layout Direction\)](#) on page 749

- [localizeText \(Localize Text\)](#) on page 751
- [location \(Location\)](#) on page 751
- [name \(Name\)](#) on page 751
- [noCheckDigits \(Number Check Digits\)](#) on page 751
- [noDigits \(Number Digits\)](#) on page 751
- [noText \(Hide Text\)](#) on page 751
- [outputOrder \(Output Order\)](#) on page 751
- [padding \(Padding\)](#) on page 752
- [pageName \(Name\)](#) on page 752
- [pageNoOffset \(Offset\)](#) on page 752
- [pageNoFormat \(Format\)](#) on page 752
- [preferRectangularSymbols \(Prefer Rectangular Symbols\)](#) on page 752
- [pWidth \(Proportional Width\)](#) on page 752
- [rawCodeValue \(Raw Code Value\)](#) on page 753
- [rBorder \(Right Border\)](#) on page 753
- [referenceDefault \(Default\)](#) on page 753
- [referenceName \(InfoNode Name\)](#) on page 753
- [rule \(Rule\)](#) on page 753
- [ruleColor \(Rule Color\)](#) on page 753
- [section \(Section\)](#) on page 754
- [scaleX \(Scale X\)](#) on page 754
- [scaleY \(Scale Y\)](#) on page 755
- [seriesTitle \(Series Title\)](#) on page 755
- [smartParse \(Smart Parse\)](#) on page 755
- [sortAscending \(Sort Ascending\)](#) on page 755
- [sortBy \(Sort By\)](#) on page 755
- [splitOversizedItem \(Split Oversized Items\)](#) on page 755
- [strikethrough \(Strikethrough\)](#) on page 756
- [swapX \(Swap X\)](#) on page 756
- [text \(Text\)](#) on page 756
- [textAlignment \(Text Alignment\)](#) on page 756
- [textExpression \(Text Expression\)](#) on page 756
- [thinToGapRelation \(Thin To Gap Relation\)](#) on page 757
- [thinToThickRelation \(Thin To Thick Relation\)](#) on page 757
- [tile](#) - replaced by [fill \(Fill\)](#) on page 746
- [title \(Title\)](#) on page 757
- [topN \(Top N\)](#) on page 758
- [trimText \(Trim Text\)](#) on page 758
- [objectType \(Type\)](#) on page 758
- [underline \(Underline\)](#) on page 758
- [URL \(Location\)](#) on page 758
- [vAlign \(Vertical Alignment\)](#) on page 758
- [value \(Value\)](#) on page 758
- [valuesTitle \(Values Title\)](#) on page 759
- [visibilityCondition \(Visibility Condition\)](#) on page 759
- [vPadding \(Vertical Padding\)](#) on page 759
- [vRule \(Vertical Rule\)](#) on page 759
- [x \(X\)](#) on page 759
- [xAxisTitle \(xAxisTitle\)](#) on page 759
- [X-Size \(X-Size\)](#) on page 759

- [X-Size Adjustment \(X-Size Adjustment\)](#) on page 760
- [y \(Y\)](#) on page 760
- [yAxisTitle \(yAxisTitle\)](#) on page 760
- [Y-Size \(Y-Size\)](#) on page 760
- [Y-Size Adjustment \(Y-Size Adjustment\)](#) on page 760

alignment (Alignment)

Controls alignment of a report element.

In the Properties view, this property is the **Alignment** property in the **Geometry** category.

Controls the **x** position of this report element in its parent container, unless you have set the **x** property explicitly.

Type: [Enum](#), the alignment choices are:

- **none** - there is no adjustment.
- **near** - shortcut for **x** = 0; that is, aligns closest to the origin of **x** within the parent container
- **far** - shortcut for **x** = **max**, **anchorX** = 1; that is, aligns the most remotely from the origin of **x** within the parent container
- **center** - shortcut for **x** = **max**/2, **anchorX** = 0.5, centered in the parent container
- **baseline** - uses baseline alignment

The default value is **none**.

See: [Placing Elements on the Report Page, Align Numbers, Center Elements](#)

anchorX (Anchor X)

Shifts the attachment point for self-adjusting nodes.

In the Properties view, this property is the **Anchor X** property in the **Geometry** category.

This property is relevant only if the property **x** is set. Shifts the attachment point for self-adjusting nodes between the point nearest to the parent's coordinate system's origin (value=0.0) and the most remote point (value=1.0). For nodes that are adjusted by their parent this attribute has no effect. A value of 0.5, for example, sets the attachment point to the center of the node.

Type: [PXML](#), point value. The default value is **0**.

anchorY (Anchor Y)

Shifts the attachment point for self-adjusting nodes.

In the Properties view, this property is the **Anchor Y** property in the **Geometry** category.

This property is relevant only if the property **y** is set. Shifts the attachment point for self-adjusting nodes between the point nearest to the parent's coordinate system's origin (value=0.0) and the most remote point (value=1.0). For nodes that are placed by their parent this attribute has no effect. A value of 0.5, for example, sets the attachment point to the center of the node.

Type: [PXML](#), point value. The default value is **0**.

baselineType (Baseline Type)

Specify which baseline of this report element should be linked to which baseline of a preceding element.

In the Properties view, this property is the **Baseline Type** property in the **Layout** category.

Provides additional information for **baseline alignment**, to specify which baseline of this report element should be linked to which baseline of a preceding element that also has the property **alignment** set to **baseline**. This property is relevant only if **alignment** for the report element is set to **baseline**.

Type [Enum](#), choices are:

- **leftleft** - the report element and its preceding element will be aligned along their left baselines
- **leftright** - the left baseline of the report element will be aligned with the right baseline of the preceding element

- `rightleft` - the right baseline of the report element will be aligned with the left baseline of the preceding element
- `rightright` - the report element and its preceding element will be aligned along their right baselines

`bBorder` (Bottom Border)

The `bBorder` property sets the weight of the bottom border of a table object.

In the Properties view, this property is the **Bottom Border** property in the **Table** category.

The `bBorder` property overrides the more general [border \(Border\)](#) on page 740 property for a table object.

Type: [PXML](#), point value.

Default value: None.

`bgColor` (Background Color)

Sets the background color for this node.

In the Properties view, this property is the **Background Color** property in the **Color** category.

The value is not inherited from the parent node.

Type: **Color**; valid colors are selected from the Edit Expression color palette. The default value is 'no background color': transparent.

`border` (Border)

The `border` property sets the weight of the border of a table object.

In the Properties view, this property is the **Border** property in the **Table** category.

Type: [PXML](#), point value.

Default value: 1

`categoryKey` (Category Key)

Specifies the key of a category in a Category Chart.

In the Properties view, this property is the **Category Key** property in the **Items** category.

Specifies the key of a category in a [Category Chart](#). Must be unique within a chart.

Type: [String](#). The default is a blank String.

`categoryTitle` (Categories Title)

Specifies the title for the categories axis in a Category Chart.

In the Properties view, this property is the **Categories Title** property in the **Chart** category.

Specifies the title for the categories axis in a [Category Chart](#).

Type: [String](#). The default is a blank String.

`check` (Check)

Check the checksum character.

In the Properties view, this property is the **Check** property in the **Bar Code** category.

When set, the checksum character of the specified code value is checked for correctness.

Type [Boolean](#). The default value is **true**.

`class` (Class)

Specifies one or more classes for a report element.

In the Properties view, this property is the **Class** property in the **Object** category.

Class names that are prefaced with the string "grw" are internally meaningful to the Genero Report Writer. The class property is available for all PXML nodes.

Example of class names with a "grw" prefix:

- grwTableHeader
- grwTableRow
- grwHeadlessTableRow
- grwTableNumericColumnValue
- grwStringValue

Type **String** (space separated list of identifiers).

clip (Clip)

Option to clip the report object box and its content along the sides.

In the Properties view, this property is the **Clip** property in the **Layout** category.

This applies to all layout nodes.

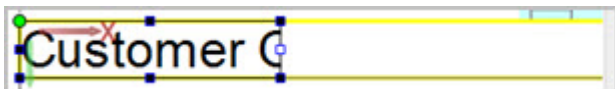


Figure 377: Clipped field

Type **Boolean**: The default value is **false**.

codeType (Code Type)

Specifies the type of Bar code.

In the Properties view, this property is the **Code Type** property in the **Bar Code** category.

This is mandatory, a default value is not set. Not all codeTypes are relevant to all Bar Codes.

The currently supported types are: [U PC-A](#), [UPC-E](#), [UPC Supplementals 2 and 5](#), [EAN-13](#), [EAN-8](#), [EAN Supplementals 2 and 5](#), [Code 128 \(EAN 128\)](#), [Code 2/5 Industrial](#), [Code 2/5 Inverted](#), [Code 2/5 IATA](#), [Code 2/5 Interleaved](#), [Code 2/5 Matrix](#), [Code 2/5 Datalogic](#), [Code BCD Matrix](#), [Code 11 Matrix](#), [Code 39](#), [Code 39 extended](#), [Code 32](#), [Code 93](#), [Code 93 extended](#), [Codabar 18](#) and [Codabar 2](#).

See [Bar Codes](#) for additional information.

Type: [Enum](#).

codeValue (Code Value)

Specifies the code value of a bar code.

In the Properties view, this property is the **Code Value** property in the **Bar Code** category.

The character set and semantic constraints depend on the code type selected.

Type: [String](#). See [Bar Codes](#) for complete information.

codeValueExpression (Code Value Expression)

In the Properties view, this property is the **Code Value Expression** property in the **Bar Code** category.

Property of the [BARCODEBOX](#), which expects a PXML string expression value to calculate a page number string. When set, the value of the property overrides the [codeValue](#) property. The value of the **codeValue** property is still used for measuring the required space, but if it is not set the default size which is computed from the expression should be sufficiently exact.

Type: [PXML](#).

If a value for this property, or for the [X-Size](#) property, is not set, a default length is used in [calculating the page number string](#) from these functions.

These functions can be used to format and access specific page numbers and totals.

- Class String: [format](#)(Numeric *number*, Enum *format*) - formats the number as specified. The value for the format parameter can be ARABIC, LOWERROMAN or UPPERROMAN.

- Class Numeric: [getPhysicalPageNumber\(\)](#) - gets the current page number of the physical page.
- Class Numeric: [getTotalNumberOfPhysicalPages\(\)](#) - gets the total number of physical pages.
- Class Numeric: [getPageNumber\(String pageName\)](#)- gets the page number of the specified page
- Class Numeric: [getTotalNumberOfPages\(String pageName\)](#) - gets the total number of pages for the specified page.

See " [Page N of M](#)" examples.

color (Color)

Sets the paint color for this node and for all children of this node.

In the Properties view, this property is the **Color** property in the **Color** category.

On [Map Charts](#) this property is used to assign each slice of the chart a specific color.

Type: Color, valid colors are selected from the Edit Expressions color palette. The default value is inherited from the parent node. The root node has the color **black**.

colspan (Column Span)

The `colspan` property is set when two or more cells are merged in a row definition for a report table.

In the Properties view, this property is the **Column Span** property in the **Table** category.

Type: whole number. The number indicates how many cells are included in the merge, starting with zero. If no cells are merged, the `colspan` would be zero; if two columns are merged, the `colspan` value is 1, meaning it spans one additional column; if three columns are merged, the `colspan` value is 2, meaning it spans two additional columns; and so on.

Default value: none.

computeAggregatesInnermostDimension (Compute aggregates on the innermost dimension)

Specifies whether or not compute aggregates on the innermost dimension

In the Properties view, this property is the **Compute aggregates on the innermost dimension** property in the **Chart** category.

Default is TRUE.

Type: [Boolean](#)

computeAverage (Compute Average)

Specifies whether or not the average should be computed for a dimension.

In the Properties view, this property is the **Compute Average** property in the **Value** category.

Default is FALSE.

Type: [Boolean](#)

computeCount (Compute Count)

Specifies whether or not the number of fact rows should be computed for a dimension.

In the Properties view, this property is the **Compute Count** property in the **Value** category.

Default is FALSE.

Type: [Boolean](#)

computeDistinctCount (Compute Distinct Count)

Specifies whether or not the number of sub elements should be computed for a dimension.

In the Properties view, this property is the **Compute Distinct Count** property in the **Value** category.

Sub elements are either sub dimensions or, in the case of the innermost dimension, fact rows.

Default is FALSE.

Type: [Boolean](#)

computeMaximum (Compute Maximum)

Specifies whether or not the maximum value should be computed for a dimension.

In the Properties view, this property is the **Compute Maximum** property in the **Value** category.

Default is FALSE.

Type: [Boolean](#)

computeMinimum (Compute Minimum)

Specifies whether or not the minimum value should be computed for a dimension.

In the Properties view, this property is the **Compute Minimum** property in the **Value** category.

Default is FALSE.

Type: [Boolean](#)

computeTotal (Compute Totals)

Specifies whether or not totals should be computed for a dimension.

In the Properties view, this property is the **Compute Totals** property in the **Value** category.

Considerations regarding chart drawing and output sorting: When selecting a chart visualization (specified by drawAs) that displays aggregated values, it is necessary that aggregation is performed on the dimensions required by the chart. Similarly, output sorting requires an aggregation function to be defined for all dimensions by which these will be sorted. In the case that more than one aggregation option is selected, the processor will pick the aggregate option that is highest up in the priority list:

1. computeTotal
2. [computeAverage](#)
3. [computeMaximum](#)
4. [computeMinimum](#)
5. [computeCount](#)
6. [computeDistinctCount](#)

Type: [Boolean](#)

Default: TRUE. Unlike the other aggregation options, totals are computed by default.

controlCharacters (Control Characters)

Configures which characters to use for textual printout of control characters in Code 93 and Code 93 extended.

In the Properties view, this property is the **Control Characters** property in the **Bar Code** category.

Code 93 defines four control characters "!?|\\" which are represented per default by the unicode characters "circled dash" '#' (229d), "circled asterisk operator" '#' (229b), "circled division slash" '#' (2298) and "circled plus" '#' (2295). Depending on the font used it might be desirable to use different characters than the default characters.

Type: String. Default value: "⊝⊛⊘⊕"

dataSymbolsPerLine (Data Symbols per Line)

Specifies the number of data symbols per line.

In the Properties view, this property is the **Data Symbols per Line** property in the **Bar Code** category.

This property is unique to the [pdf-417](#) on page 795 bar code type.

Type: Integer value.

Specifies the number of data symbols per line. The value must be an integer between 1 and 30. Low values cause more narrow printout with more lines. The number of lines is not allowed to exceed 90. It should be noted, that the overall required image space usually grows with lower values because there is a constant amount of organizational information which is added with each additional line. This is not

generally the case, since lines have to be filled with padding so that specially with small amounts of data a larger value may actually create a larger image. If the value is not specified, the system computes a value that minimizes image space.

Fails if: Value cannot be parsed as a integer value. Value is not in the range 1...30.

Default value: A value that minimizes the overall image size.

displayFactRows (Display Fact Rows)

Specifies whether or not fact rows (the individual unaggregated data items) are displayed.

In the Properties view, this property is the **Display Fact Rows** property in the **Chart** category.

This is applicable only if the selected output visualization (specified by [drawAs](#)) is capable of drawing individual rows. This is currently the case for the "Table" visualization type only.

Type: [Boolean](#)

displayRecurringDimensions (Display Recurring Dimension Values)

Specifies whether or not recurring dimension values in the same column of table output should be displayed.

In the Properties view, this property is the **Display Recurring Dimension Values** property in the **Chart** category.

By default, cells with recurring values are left empty.

Type: [Boolean](#)

displaySelection (Display Selection)

Selects which of the declared dimensions or measures should be displayed.

In the Properties view, this property is the **Display Selection** property in the **Chart** category.

For example, given a table with 4 dimensions, specifying a value of "3,2,0" selects the last, the second last and the first column for display.

Depending on the visualization type (set by the [drawAs](#) property), it is possible that not all selected dimensions will display.

Not specifying a value is equivalent to selecting all declared dimensions. For example, given a table with three dimensions, not specifying this attribute is the equivalent of specifying a value of "0,1,2".

For dimensions, specifying an empty set will display the measures only and the grand total line.

For measures, specifying an empty set will display the dimensions, their aggregates and the grand total line.

Type: [Column selector](#)

drawAs (Draw As)

Specifies the type of chart that is rendered from the data.

In the Properties view, this property is the **Draw As** property in the **Chart** category.

This property also allows you to specify that the chart [displays as a table](#).

Type: [Enum](#).

Valid values for Category Chart: Bar, Bar3D, Stacked Bar, Line, Line3D, Area, Stacked Area, or Waterfall, Table, Sorted Table, Aggregated Table

Note: If you select **Waterfall** as the chart type, the value in the last category of the data set should be (redundantly) specified as the sum of the items in the preceding categories - otherwise, the final bar in the chart will be incorrectly plotted. At the present time, the chart can only have one category.

Valid values for Map Chart: Pie, Pie3D, Bar, Bar3D or Ring, Table, Sorted Table, Aggregated Table

Valid values for XY Chart: Polar, Scatter, Area, Stacked Area, Line, Step, Step Area, or Time Series, Table, Sorted Table

Valid values for Pivot Table: Pie, Pie3D, Ring, Bar, Bar3D, Area, StackedBar, StackedArea, Line, Line3D, Waterfall, Polar, Scatter, XYArea, XYStackedArea, XYLine, Step, StepArea, TimeSeries, Table

drawLabels (Draw Labels)

Controls whether Labels are drawn for a Map Chart.

In the Properties view, this property is the **Draw Labels** property in the **Chart** category.

Controls whether the Labels are drawn for a [MapChart](#).

Type: [Boolean](#). The default value is **true**.

drawLegend (Draw Legend)

Controls whether the Legend is drawn for a Map Chart.

In the Properties view, this property is the **Draw Legend** property in the **Chart** category.

Controls whether the Legend is drawn for a [MapChart](#). The option to remove the legend is useful when more than several charts are drawn next to each other in a document. You can make the charts share a single legend by specifying the legend only on one of the charts.

Type: [Boolean](#). The default value is **true**.

encoding (Encoding)

Sets the encoding for non ascii characters in the code value.

In the Properties view, this property is the **Encoding** property in the **Bar Code** category.

This property is unique to the [pdf-417](#) on page 795 bar code type.

Type: Encoding

Sets the encoding for non ascii characters in the code value. Run "`java CharsetInfo`" for a list of character set encodings available on a particular platform. Valid example values are 'ISO-8859-15' or 'IBM437'.

Fails if:

- Value is not a valid host name
- Socket connection cannot be established

Default value: not set (the lower 8 bits of the unicode values are encoded)

enumValues (Enum Values)

Specifies an optional list of strings that represent ordinal values.

In the Properties view, this property is the **Enum Values** property in the **Value** category.

This attribute is applicable for numeric dimensions only whose value is limited to a range of whole numbers representing a set of symbols.

Consider a dimension containing the values 0 through 11, representing the months of the year (0=Jan, 1=Feb, ..., 11=Dec). For this example, you could set the enumValues = "Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sept, Oct, Nov, Dec". When the dimensions are sorted ([inputOrder](#) and [displaySelection](#) differ), it can make a visual difference whether the column is declared a numeric enumeration or as a string column containing the literal value and not its ordinal value. The difference comes from the sorting. In the first case (a numeric enumeration), the month will be displayed in the order "Jan, Feb, ..., Dec". In the second case, the alphabetic order of the month names would result in the order "Apr, Aug, Dec, ...".

Type: List of quotable strings

errorCorrectionDegree (Error Correction Degree)

Specifies the error correction degree.

In the Properties view, this property is the **Error Correction Degree** property in the **Bar Code** category.

This property is unique to the [pdf-417](#) on page 795 bar code type.

Type: Integer value.

Specifies the error correction degree. Valid values are in the range 0...8. Higher values make the image more robust.

Fails if: Value cannot be parsed as a integer value. Value is not in the range 0...8.

Default value: A value that proportional to the data size

fidelity (Text Fidelity)

Controls the way text is output.

In the Properties view, this property is the **Text Fidelity** property in the **Font** category.

When set, this property ensures that the preview and printout are 100% the same. In some cases this is necessary when the operating system font definitions deviate from the built-in font definitions in the printer. This flag then instructs the output routine not to use the printer font.

Type: [Boolean](#). The default value is **false**.

fill (Fill)

Specifies how an image fills an area.

In the Properties view, this property is the **Fill** property in the **Image** category.

Replaces the previous property **tile**. When a document containing the tile property is opened in version 2.4x or higher, it is automatically replaced with this property.

This property is relevant when both [x-Size](#) and [y-Size](#) are set so that the outer image bounds are defined.

Type: [Enum](#). The value can be one of:

- **completely** - The images is resized to fill the specified area. The aspect ratio is not respected.
- **preserveAspectRatio** - The images is resized to fit the specified area. The aspect ratio is respected. Setting the attributes [X-Size_Adjustment](#) and [Y-Size_Adjustment](#) to a value of "shrinkToChildren" will shrink the final bounds of the box so that it has the same size as the image.
- **tile** - The image is painted as tiles into the specified area.
- **clip** - The image is painted without scaling, and it is clipped at the edges of the specified area if it is too large to fit.

floatingBehavior (Floating Behavior)

Controls the sizing behavior of the parent box when this node floats (sets x or y).

This property is not relevant if x or y are not set.

In the Properties view, this property is the **Floating Behavior** property in the **Layout** category.

The valid values are;

- **enclosed** - will make the parent size itself so that this object will be enclosed in the parent. All objects dragged from the toolbox or Data View have this property set to **enclosed**. Note that the node can still float outside of the parent using negative values for x or y. A
- **free** - will make the parent ignore this node during sizing, allowing it to float outside of its bounds.

Type: [Enum](#). Valid values are **free** or **enclosed**.

fontBold (Bold)

Sets a **bold** font style for this node and for all children of this node.

In the Properties view, this property is the **Bold** property in the **Font** category.

The root node font style is **plain**.

Type: [Boolean](#). The default value is inherited from the parent.

fontItalic (Italic)

Sets an **italic** font style for this node and for all children of this node.

In the Properties view, this property is the **Italic** property in the **Font** category.

The root node font style is **plain**.

Type: [Boolean](#). The default value is inherited from parent.

fontName (Name)

Sets the font face name for this node and for all children of this node.

In the Properties view, this property is the **Name** property in the **Font** category.

Type: [Enum](#), platform-dependent. The default value is inherited from the parent. The root node has a platform-dependent Sans Serif font.

See the Genero Studio >> Report Writer documentation topic "Configuring Fonts and Printers" for additional information about fonts.

fontSize (Size)

Sets the font size in points for this node and for all children of this node.

In the Properties view, this property is the **Size** property in the **Font** category.

During expression evaluation the variable **fontSize** is available, which contains the inherited font size. This variable can be used for relative font sizing. For example, the expression `fontSize="fontSize*1.2"` makes the current font 20% larger than the parent font.

Type: [Numeric](#). The default value (not the expression) is inherited from the parent. The root node has a font size of 12 point.

format (Format)

Specify (control) the output of a numeric display.

In the Properties view, this property is the **Format** property in the **Value** category.

For DECIMAL data types, *format-string* consists of pound signs (#) that represent digits and a decimal point. For example, "###.##" produces three places to the left of the decimal point and exactly two to the right.

See [Align and Format Numbers](#). See the BDL **format** attribute for additional information and format characters.

Type: [String](#).

fWidth (Fix Width)

Sets the fixed width of a column.

In the Properties view, this property is the **Fix Width** property in the **Table** category.

When you specify a value as Fix Width, you are giving an absolute size for that column. By default, the number entered refers to points, but you can change the unit of measure by specifying the type of units used. See [Unit Names](#) on page 801.

Type: [PXML](#), point value.

Default value: None.

hAlign (Horizontal Alignment)

The `hAlign` property defines the horizontal alignment of a value in its cell for all cells in a column of a report table.

In the Properties view, this property is the **Horizontal Alignment** property in the **Table** category.

Type: [Enum](#), the alignment choices are:

- **left** - Left-justify in the column.
- **right** - Right-justify in the column.
- **center** - Center in the column
- **baseline** - uses baseline alignment

The default value is **left**.

hidePageHeaderOnLastPage (Hide Page Header On Last Page)

Suppresses the drawing of beforeFirst, firstPageHeader, evenPageHeader and oddPageHeader material on the last page.

In the Properties view, this property is the **Hide Page Header On Last Page** property in the **Mini Page** category.

Child elements querying for available space will, however, be given space values that are reduced by the size of the header as if it were drawn. Type: [Boolean](#). The default value is **false**.

See also: [Page Headers and Footers](#)

hidePageFooterOnLastPage (Hide Page Footer On Last Page)

Suppresses the drawing of afterLast, firstPageTail, evenPageTail and oddPageTail material on the last page.

In the Properties view, this property is the **Hide Page Footer On Last Page** property in the **Mini Page** category.

Child elements querying for available space will, however, be given space values that are reduced by the size of the footer as if it were drawn. Type: [Boolean](#). The default value is **false**.

See also: [Page Headers and Footers](#)

hierarchiesInputOrder (Hierarchies input order)

Specifies the order by which the data is presorted.

In the Properties view, this property is the **Hierarchies input order** property in the **Chart** category.

If nothing is specified, the data is assumed to be presorted in the declaration order of the dimensions. This is the default. For example, given a table with three dimensions, not specifying this attribute is the equivalent of specifying a value of "0,1,2".

Specifying an empty set indicates that the data will arrive unsorted. Large data amounts should be at least partially presorted.

Specifying a wrong input order can cause runtime errors and yield incorrect results.

Type: [Order specifier](#)

href (href)

This property can be used to define a hyperlink pointing to any resource on the Internet, local disk, or to any anchor inside the report document.

In the Properties view, this property is the **href** property in the **Hyperlink** category.

See [id](#) for additional information about creating anchors. The **href** should be defined using the URI syntax. See [Using Hyperlinks in a Report](#).

Type: [String](#).

hPadding (Horizontal Padding)

The `hPadding` property sets the width of the horizontal padding for a column in a table object.

In the Properties view, this property is the **Horizontal Padding** property in the **Table** category.

The `hPadding` property overrides the value set for the [padding \(Padding\)](#) on page 752 property when setting the vertical padding for a table in a report.

Type: [PXML](#), point value.

Default value: None.

hRule (Horizontal Rule)

The `hRule` property controls the width of the horizontal rule lines for a table. Horizontal rule lines separate rows.

In the Properties view, this property is the **Horizontal Rule** property in the **Table** category.

The property `hRule` overrides the [rule \(Rule\)](#) on page 753 property.

Type: [PXML](#), point value.

Default value: none.

id (id)

This property can be used to create an anchor in the report document.

In the Properties view, this property is the **id** property in the **Hyperlink** category.

Nodes can be identified with a unique **id** and then used as the target of an [href](#) hyperlink. See [Using Hyperlinks in a Report](#).

Type: [String](#).

indent (Indent)

Specifies the indentation value for the paragraph.

In the Properties view, this property is the **Indent** property in the **Text** category.

The value may be negative.

Type: [PXML](#). The default value is 0.

intendedResolution (Intended Resolution)

Controls the mapping of pixels to device pixels.

In the Properties view, this property is the **Intended Resolution** property in the **Image** category.

This property is relevant only if neither [X-Size](#) nor [Y-Size](#) are set, or if **tile** is set.

The default is the current screen resolution.

key (Key)

Specifies the key of the item in a chart.

In the Properties view, this property is the **Key** property in the **Items** category.

Within a [category chart](#), specifies the key of an item within a category; must be unique.

Type: [String](#).

keysTitle (Keys Title)

Specifies the title of the keys Axis (usually the y Axis) of a Business Chart.

In the Properties view, this property is the **Keys Title** property in the **Chart** category.

Type: [String](#). The default is a blank String.

layoutDirection (Layout Direction)

The `layoutDirection` property controls the direction in which child elements are laid out, which is also the direction of the Y-axis.

In the Properties view, this property is the **Layout Direction** property in the **Orientation** category.

Choices are:

- `topToBottom`
- `leftToRight`

- bottomToTop
- rightToLeft
- unturned
- turnRight
- upsideDown
- turnLeft
- inherit - the element inherits the orientation of its parent
- swapped - if the [swapX](#) property is also set to **swapped**, the element inherits the swapped orientation of its parent.

These values depends on the language of the system where GRE is running; this allows you to have the layoutDirection follow the custom of the language.

- horizontalNatural - follows the custom of the system language
- horizontalUnnatural - reverses the natural order of the system language
- verticalNatural - follows the custom of the system language
- verticalUnnatural - reverses the natural order of the system language*

For example, to have a horizontal layout that is leftToRight when the language is European, but rightToLeft for Arabic, select **horizontalNatural**. Select **horizontalUnnatural** to reverse the natural order of the language in the horizontal layout.

Type: [Enum](#). The default value is **topToBottom**. By default the positive X-axis extends 90 degrees right (clockwise) of the positive Y-axis.

Layout direction and the graphical report designer

The parent container of the currently focused item is highlighted by a dashed, slowly moving yellow border. The border moves in the layout direction and is open at the far side of the layout direction forming a “U” shape.



Figure 378: A top-to-bottom layout direction

In this example, the layout direction is top-to-bottom (as illustrated by the arrow). The box is not closed at the bottom.

lBorder (Left Border)

The `lBorder` property sets the weight of the left border of a table object.

In the Properties view, this property is the **Left Border** property in the **Table** category.

The `lBorder` property overrides the more general [border \(Border\)](#) on page 740 property for a table object.

Type: [PXML](#), point value.

Default value: None.

localizeText (Localize Text)

Check the box to indicate that there is a localized string for this value.

In the Properties view, this property is the **Localize Text** property in the **Text** category.

Checking the box enables the localization of text contents in [Word Boxes](#) and [WordWrap Boxes](#).

Type: [Boolean](#). Valid choices are True, False. The default value is False.

location (Location)

The location of an image file.

In the Properties view, this property is the **Location** property in the **Image** category.

Location values are URLs supporting the protocols "http", "file" and "data".

The location can be set to an absolute path and filename. In this instance, the image file remains the same for the duration of the report.

Variables (RTL expressions) can be used if the image file will change during processing, such as when the image file name is stored in the database and the value can change for each row processed. You specify a URL based on one or more variables using an RTL expression. This is demonstrated in the "OrderReport.4rp" where the "ImageBox2" element has the expression ". /images / database / "+orderline.product.prodpic.trim()"

Type: [String](#).

name (Name)

Assigns the specified name to the node for debugging purposes.

In the Properties view, this property is the **Name** property in the **Object** category.

The name must be unique in the document.

Type: [String](#). The default value is "".

noCheckDigits (Number Check Digits)

Controls the expected number of check digits for codeValue for codes of type "code-11-matrix" and "code-93".

In the Properties view, this property is the **Number Check Digits** property in the **Bar Code** category.

Type: [Numeric](#).

noDigits (Number Digits)

Controls the expected number of digits for codeValue for a code type that allows a variable number of digits.

In the Properties view, this property is the **Number Digits** property in the **Bar Code** category.

Specifically these are the code types "code-2-5-industrial", "code-2-5-inverted", "code-2-5-IATA", "code-2-5-interleaved", "code-2-5-matrix", "code-2-5-datalogic", "code-BCD-matrix", "code-11-matrix", "code-39", "code-39-extended", "codabar-18" and "codabar-2".

Type: [Numeric](#).

noText (Hide Text)

Suppresses text output if set.

In the Properties view, this property is the **Hide Text** property in the **Bar Code** category.

Type: [Boolean](#). The default value is **false**.

outputOrder (Output Order)

Specifies the order by which the data should be presented.

In the Properties view, this property is the **Output Order** property in the **Chart** category.

Not specifying a value or specifying the empty set will output the data in the order it was received.

Consider a pivot table with the dimensions "country" and "region" and the measure "turnover". Specifying an output order of "-1" creates an output in which the country with the highest turnover is listed first. Within that country the region with the highest turnover is listed first and within each country the individual fact rows are ordered in descending order by the turnover value.

Note: Specifying an output order may cause large latency and memory consumption on large input data.

When an output order is specified, it is possible to specify a cutoff value called [topN](#) that limits the number of items displayed. Continuing with our example, a cutoff value of 2 would limit the output to the two top countries; within each country the two top regions; and within each region to the two highest fact rows.

Note: Specifying an output order currently limits the number of displayable aggregations to one per dimension. If more are specified, one is picked following a priority list. See [computeTotal](#).

Type: [Order specifier](#)

padding (Padding)

Sets the width of all of an object's padding.

In the Properties view, this property is the **Padding** property in the **Table** category.

Can be overridden by specific padding properties.

Type: [PXML](#), point value.

Default value: None.

pageName (Name)

Name of a parent node.

In the Properties view, this property is the **Name** property in the **Page Number** category.

Type: [String](#). No default value is set.

pageNoOffset (Offset)

An offset that is added to the current page number.

In the Properties view, this property is the **Offset** property in the **Page Number** category.

When set to 100, for example, the first page will be number 101.

Type: [Numeric](#). The default value is **0**.

pageNoFormat (Format)

Sets the number format type.

In the Properties view, this property is the **Format** property in the **Page Number** category.

Type: [Enum](#). Valid values are Arabic, Lowerroman, Upperroman. The default value is **Arabic**.

preferRectangularSymbols (Prefer Rectangular Symbols)

Enables rectangular symbols.

In the Properties view, this property is the **Prefer Rectangular Symbols** property in the **Bar Code** category.

The [data-matrix](#) bar code is usually quadratic, and any code value can be represented by a quadratic symbol. If you are concerned about running out of space in the vertical of the page, you might prefer a symbol that is wider than it is high. Check the box to enable rectangular symbols.

Type: [Boolean](#). Valid choices are True, False. The default value is False.

pWidth (Proportional Width)

Sets the proportional width of a column.

In the Properties view, this property is the **Proportional Width** property in the **Table** category.

When you specify a value in the Proportional Width property, you are specifying its width in proportion to other columns in the same table. Consider the following example: A table has three columns: A, B and C. Column A has a proportional width setting of 1, column B has a proportional width of 2, and column C has a proportional width of 3. This means that column B is two times as wide as column A, and column C is three times as wide as column A.

Type: [PXML](#), point value.

Default value: None.

rawCodeValue (Raw Code Value)

Specify the code value at a lower level than `codeValue`.

In the Properties view, this property is the **Raw Code Value** property in the **Bar Code** category.

This property is unique to the [pdf-417](#) on page 795 bar code type.

Type: A comma-separated list of integers in the range 0...899.

This attribute can be used instead of `codeValue` to specify the code value at a lower level giving more control on the encoded data.

Fails if: Encoding for non-ASCII characters in the code value.

Default value: not set

rBorder (Right Border)

The `rBorder` property sets the weight of the right border of a table object.

In the Properties view, this property is the **Right Border** property in the **Table** category.

The `rBorder` property overrides the more general [border \(Border\)](#) on page 740 property for a table object.

Type: [PXML](#), point value.

Default value: None.

referenceDefault (Default)

The text value to be displayed, if the reference cannot be resolved.

In the Properties view, this property is the **Default** property in the **Reference** category.

Type: [String](#). The default value is "-" .

referenceName (InfoNode Name)

The name of the Info Node referenced.

In the Properties view, this property is the **InfoNode** property in the **Reference** category.

The name of the [InfoNode](#) that is referenced.

Type: [String](#). Mandatory; there is no default value.

rule (Rule)

The `rule` property sets the weight of the line between two rows or two columns in a table object.

In the Properties view, this property is the **Rule** property in the **Table** category.

The properties [hRule \(Horizontal Rule\)](#) on page 749 and [vRule \(Vertical Rule\)](#) on page 759 can override the default value set by the `rule` property for a table object.

Type: [PXML](#), point value.

Default value: 1

ruleColor (Rule Color)

The `ruleColor` property controls the color of the rule for a table.

In the Properties view, this property is the **Rule Color** property in the **Table** category.

Type: [The Color Class](#) on page 807

Default value: none.

section (Section)

The layout node attribute that controls printing within a parent MiniPage.

In the Properties view, this property is the **Section** property in the **Layout** category.

This attribute of a Layout Node object specifies that the content of the node should print on its parent [MiniPage](#) at the location specified by this property.

Type: [Enum](#).

When you select a MiniPage for a page header or footer, for example, you specify where the container should be printed on its parent [MiniPage](#) by setting the section property. The report output prints any headers and footers that you have set, based on priorities of each of these values:

firstPageHeader	The Page Header to be printed on the first page only; if this section is defined, subsequent Page Headers begin printing on the second page.
anyPageHeader	The Page Header for every page, unless separate Odd and Even Page Headers are defined.
oddPageHeader	Specific Page Header to be printed on odd pages.
evenPageHeader	Specific Page Header to be printed on even pages.
firstPageFooter	A Page Footer that prints on the first page only; if this section is defined, subsequent page footers begin printing on the second page.
anyPageFooter	The Page Footer for every page, unless separate Odd and Even Page Footers are defined.
oddPageFooter	Specific Page Footer to be printed on odd pages.
evenPageFooter	Specific Page Footer to be printed on even pages.
lastPageFooter	A Page Footer that prints on the last page only. For the last page, this node takes priority over oddPageFooter, evenPageFooter or anyPageFooter nodes. A node with this section value set must be located as the last in the sibling list.
itemSeparator	Prints between each sibling element, as long as there is more than one element.

If you have an anyPageHeader and a firstPageHeader, for example, the anyPageHeader content will print only on the second and subsequent pages.

Important: If you set the section property for any node, the node may not be preceded in its sibling list in the Report Structure tree by any nodes that do not have this property set.

A layout node with a defined section attribute is also known as a *named port*. A layout node without a defined section attribute is also known as a *primary port*.

See [Print Headers and Footers](#).

scaleX (Scale X)

Applies the specified scale in this x-direction.

In the Properties view, this property is the **Scale X** property in the **Geometry** category.

The scale affects everything contained in this node (children, children-children, etc.). Scales are cumulative.

Type: [PXML](#). The default value is **1.0**.

scaleY (Scale Y)

Applies the specified scale in this y-direction.

In the Properties view, this property is the **Scale Y** property in the **Geometry** category.

The scale affects everything contained in this node (children, children-children, etc.). Scales are cumulative.

Type: [PXML](#). The default value is **1.0**.

seriesTitle (Series Title)

Title of the series in an XY Chart.

In the Properties view, this property is the **Series Title** property in the **Items** category.

Type: [String](#) .

smartParse (Smart Parse)

Controls the parsing of Bar Code Boxes.

In the Properties view, this property is the **Smart Parse** property in the **Bar Code** category.

Specifies that the [codeValue](#) property for [Bar Code Boxes](#) is parsed in "smart" mode. By default it is parsed in raw mode.

In "smart" mode the codeValue is interpreted literally. An attempt is made to map the characters contained in the string to one or more codes choosing the shortest possible representation. Control characters cannot be displayed in this mode. Currently the functionality is only available for "code-39-extended".

Type: [Boolean](#). The default value is **false**.

sortAscending (Sort Ascending)

Sorts the values in ascending order.

In the Properties view, this property is the **Sort Ascending** property in the **Chart** category.

Set this value to false to reverse the display order specified by the sortBy property.

Type: [Boolean](#).

sortBy (Sort By)

Specifies the order in which the items of a chart are displayed.

In the Properties view, this property is the **Sort By** property in the **Chart** category.

Valid values are:

- Key - Sort alphabetically by the key value (MapChart) or by the category/key value (CategoryChart).
- Value - Sort by the numeric value.
- InputOrder - Preserve the order in which the items were defined. If more than one value is received for a particular key value (MapChart) or category/key value combination (CategoryChart), the first value received defines the order.

By setting the property sortAscending to false a reverse sorting for each of these options can be obtained.

Type: Enumeration.

splitOversizedItem (Split Oversized Items)

Defines the behavior for when a single item exceeds the space in layout direction.

In the Properties view, this property is the **Split Oversized Items** property in the **Mini Page** category.

When set to TRUE, this value allows the splitting of large non-propagating items (such as HTMLBOX, WORDWRAPBOX or IMAGEBOX) into chunks using preferable breakpoints (if available). Preferable breakpoints refer to whitespace or between table rows.

Otherwise the box becomes overfull.

Note: The splitting of a large item is a costly operation. The item that is split is kept in memory until the last split has been performed. The item that is split should not exceed a few pages.

Type: [Boolean](#).

strikethrough (Strikethrough)
Specifies strikethrough for the text.

In the Properties view, this property is the **Strikethrough** property in the **Font** category.

Type: [Boolean](#). The default value is **false**.

swapX (Swap X)
Reverses the direction of the X-axis.

In the Properties view, this property is the **Swap X** property in the **Orientation** category.

By default the positive X-axis extends 90 degrees right (clockwise) of the positive Y-axis; if, for example, the Y-Axis points to north, the X axis will point eastward. Setting this value reverses the direction so that (in the example) the X-Axis would point to the west which is 90 degrees to the left (counter clockwise) of the Y-Axis.

Type: [Boolean](#). The default value is **false**.

text (Text)
The `text` property specifies the text to be drawn.

In the Properties view, this property is the **Text** property in the **Text** category.

Occurrences of the newline character "\n" within the string cause line breaks.

Type: [String](#). The default value is "".

For [Word Boxes](#), [WordWrap Boxes](#), and [Decimal Format Boxes](#), the value of this property may be edited directly in the report design document instead. Double-click the object and the input cursor will be placed in the text. The layout of the document is updated on each keystroke.

For [PageNo Boxes](#), [BarCode Boxes](#), and [Reference Boxes](#), the value of the text property specifies the maximum width.

textAlignment (Text Alignment)
Controls the horizontal alignment of text.

In the Properties view, this property is the **Text Alignment** property in the **Text** category.

Type: [Enum](#). Values are left, center, right, justified. The default value is **left**.

textExpression (Text Expression)
A PXML string expression value to calculate a page number string.

In the Properties view, this property is the **Text Expression** property in the **Text** category.

Property of the [PAGENOBOX](#), which expects a PXML string expression value to calculate a page number string. When set, the value of the property overrides all other formatting relevant properties of the [PAGENOBOX](#), although the [text](#) property is still used to set the width.

Type: [PXML](#).

If a value for this property, or for the [X-Size](#) or [text](#) properties, is not set, a default length is used in [calculating the page number string](#) from these functions:

These functions can be used to format and access specific page numbers and totals.

- Class String: [format](#)(Numeric *number*, Enum *format*) - formats the number as specified. The value for the format parameter can be ARABIC, LOWERROMAN or UPPERROMAN.
- Class Numeric: [getPhysicalPageNumber](#)() - gets the current page number of the physical page.
- Class Numeric: [getTotalNumberOfPhysicalPages](#)() - gets the total number of physical pages.
- Class Numeric: [getPageNumber](#)(String *pageName*)- gets the page number of the specified page
- Class Numeric: [getTotalNumberOfPages](#)(String *pageName*) - gets the total number of pages for the specified page.

See " [Page N of M](#)" examples.

thinToGapRelation (Thin To Gap Relation)

Controls the ratio of thin bars to the gaps between individual digits.

In the Properties view, this property is the **Thin To Gap Relation** property in the **Bar Code** category.

The value of a gap is calculated by the formula $GAPWIDTH=THINBARWIDTH/thinToGapRelation$. This parameter applies only to the code types "code-2-5-industrial", "code-2-5-inverted", "code-2-5-IATA", "code-2-5-interleaved", "code-2-5-matrix", "code-2-5-datalogic", "code-BCD-matrix", "code-11-matrix", "code-39", "code-39-extended", "code-32", "codabar-18" and "codabar-2".

Type: [Numeric](#). Default values: 0.5 for the types "code-2-5-industrial", "code-2-5-inverted" and "code-2-5-IATA"; 1 for the types "code-2-5-interleaved", "code-2-5-matrix", "code-2-5-datalogic" "code-BCD-matrix", "code-11-matrix", "code-39", "code-39-extended", "code-32", "code-93" and "code-93-extended".

thinToThickRelation (Thin To Thick Relation)

Controls the ratio of thin bars to thick bars.

In the Properties view, this property is the **Thin To Thick Relation** property in the **Bar Code** category.

The value of a thick bar is calculated using the formula $THICKBARWIDTH=THINBARWIDTH/thinToThickRelation$. This parameter applies only to the [code types](#) "code-2-5-industrial", "code-2-5-inverted", "code-2-5-IATA", "code-2-5-interleaved", "code-2-5-matrix", "code-2-5-datalogic", "code-BCD-matrix", "code-11-matrix", "code-39", "code-39-extended", "code-32". and "codabar 2".

Type: [Numeric](#). Default values: 1/3

title (Title)

Specifies the title of the report, output, or column.

In the Properties view, this property is the **Title** property in the **Metadata** category.

For a report, specifies the metadata for the title of the report. In the case of SVG, the title property is used as a document caption in Genero Report Viewer.

For a pivot table, specifies the title of the output. If and where this text is rendered depends on the selected visualization type (specified by [drawAs](#)).

For a pivot table hierarchy and measure, specifies the title of the column. If and where this text is rendered depends on the selected visualization type (specified by [drawAs](#)).

Type: [String](#).

tBorder (Top Border)

The `tBorder` property sets the weight of the top border of a table object.

In the Properties view, this property is the **Top Border** property in the **Table** category.

The `tBorder` property overrides the more general [border \(Border\)](#) on page 740 property for a table object.

Type: [PXML](#), point value.

Default value: None.

topN (Top N)

Specifies the number of records to display.

In the Properties view, this property is the **Top N** property in the **Chart** category.

Only valid when **outputOrder** is specified. The specified value limits the number of distinct dimension values displayed for each dimension and the number of fact rows displayed for the innermost dimension to the specified number.

Type: Integer

trimText (Trim Text)

Controls the trimming of spaces.

In the Properties view, this property is the **Trim Text** property in the **Text** category.

Controls the trimming of spaces of the value of the **text** attribute.

Type: **Enum**. Values are both, compress, left, right. The default value is not set.

transformTransparently (Transform transparently)

The **transformTransparently** property changes the effect of the properties **layoutDirection** and **swapX**. When set, the transformation extends to the entire fragment so that entire documents can be rotated.

In the Properties view, this property is the **Transform transparently** property in the **Orientation** category.

Type: **Boolean**.

objectType (Type)

The type of the report element.

In the Properties view, this property is the **Type** property in the **Object** category.

This is automatically set when you drop a specific element on the page.

underline (Underline)

Specifies that the text is underlined.

In the Properties view, this property is the **Underline** property in the **Font** category.

Type: **Boolean**. The default value is **false**.

URL (Location)

Specifies the loading location or the name of the image to display.

In the Properties view, this property is the **Location** property in the **Image** category.

Type: **String**. A value is mandatory; there is no default value.

vAlign (Vertical Alignment)

The **vAlign** property defines the vertical alignment of a value in its cell for all cells in a column of a report table.

In the Properties view, this property is the **Vertical Alignment** property in the **Table** category.

Type: **Enum**, the alignment choices are:

- **left** - Left-justify in the column.
- **right** - Right-justify in the column.
- **center** - Center in the column
- **baseline** - uses baseline alignment

The default value is **top**.

value (Value)

Specifies the value of the item.

In the Properties view, this property is the **Value** property in the **Miscellaneous** category.

Type (non-pivot table): [Numeric](#).

Type (pivot table): Can be a String or Float depending on the declared [dataType](#). In case of a numeric column, this value is converted to a float value. The entered value will fail if:

- The value is not set.
- The column is declared as numeric and the value cannot be parsed as a float point value.

valuesTitle (Values Title)

Specifies the Title of the values axis of a Business chart.

In the Properties view, this property is the **Values Title** property in the **Chart** category.

Type: [String](#).

visibilityCondition (Visibility Condition)

Boolean value (TRUE/FALSE) indicating whether the object is visible (not hidden).

In the Properties view, this property is the **Visibility Condition** property in the **Object** category.

Type: [Boolean](#). The default is TRUE.

vPadding (Vertical Padding)

The `vPadding` property sets the width of the vertical padding for a column in a table object.

In the Properties view, this property is the **Vertical Padding** property in the **Table** category.

The `vPadding` property overrides the value set for the [padding \(Padding\)](#) on page 752 property when setting the vertical padding for a table in a report.

Type: [PXML](#), point value.

Default value: None.

vRule (Vertical Rule)

The `vRule` property controls the width of the vertical rule lines for a table. Vertical rule lines separate columns.

In the Properties view, this property is the **Vertical Rule** property in the **Table** category.

The property `vRule` overrides the [rule \(Rule\)](#) on page 753 property.

Type: [PXML](#), point value.

Default value: none.

x (X)

Specifies the x-value of a X/Y coordinate pair defined by the element.

In the Properties view, this property is the **X** property in the **Geometry** category.

The x value is an offset in the [X-Size](#) direction of the parent.

Type: [Numeric](#). The default value is calculated during placing by the parent.

xAxisTitle (xAxisTitle)

Specifies the title of the x Axis of a Business Chart.

In the Properties view of an XY chart, this property is the **xAxisTitle** property in the **Chart** category.

Type: [String](#).

X-Size (X-Size)

Gives the box a fixed dimension.

In the Properties view, this property is the **X-Size** property in the **Geometry** category.

Note: If you want to preserve the aspect ratio of an image, set the value of either [Y-Size](#) or X-Size only, and allow Report Writer to calculate the corresponding value. If you set both properties, the resulting image will appear distorted.

Type: [Numeric](#). The default value is calculated after the node has completed child alignment. The value is set to the smallest possible value that encloses all children without clipping any of them.

X-Size Adjustment (X-Size Adjustment)

Specifies how the adjustment to the X-Size is to be made.

In the Properties view, this property is the **X-Size Adjustment** property in the **Geometry** category.

A value of **shrinktoChildren** shrinks the [X-Size](#) as much as possible without clipping any of the children. A value of **expandToParent** causes the box to stretch as much as possible.

For objects that draw output with a defined size (Word Boxes, BarCode Boxes, Image Boxes where the tile property is set to FALSE), the value **shrinkToChildren** will not shrink the object below this size even if all children are smaller than this size or there are no children at all. Note that self-placing children are not considered.

Type: [Enum](#). Choices are **shrinkToChildren**, **expandToParent**.

y (Y)

Specifies the y-value of a X/Y coordinate pair defined by the element.

In the Properties view, this property is the **Y** property in the **Geometry** category.

Changing the value adjusts the node at the specified coordinate. The coordinate value is an offset in the [Y-Size](#) direction of the parent.

Type: [Numeric](#). The default value is calculated during placing by the parent.

yAxisTitle (yAxisTitle)

Specifies the title of the y Axis of a Business Chart.

In the Properties view of an XY chart, this property is the **yAxisTitle** property in the **Chart** category.

Type: [String](#).

Y-Size (Y-Size)

Gives the box a fixed dimension.

In the Properties view, this property is the **Y-Size** property in the **Geometry** category.

Note: If you want to preserve the aspect ratio of an image, set the value of either Y-Size or [X-Size](#), allowing Report Writer to calculate the corresponding value. If you set both properties, the resulting image will appear distorted.

Do not set a value for this property in [WordWrapBoxes](#), because the element should typically grow based on its content.

Type: [Numeric](#). The default value is calculated after the node has completed its child alignment. The value is set to the smallest possible value that encloses all children without clipping any of them.

Y-Size Adjustment (Y-Size Adjustment)

Specifies how the adjustment to the Y-Size is to be made.

In the Properties view, this property is the **Y-Size Adjustment** property in the **Geometry** category.

A value of **shrinktoChildren** shrinks the length of [y](#) as much as possible without clipping any of the children. A value of **expandToParent** causes it to stretch as much as possible.

For objects that draw output with a defined size (Word Boxes, BarCode Boxes, Image Boxes where the tile property is set to FALSE), the value **shrinkToChildren** will not shrink the object below this size even if all children are smaller than this size or there are no children at all. Note that self placing-children are not considered.

Type: [Enum](#). Choices are **shrinkToChildren**, **expandToParent**.

Properties related to margins and borders

Margin and Border-related properties for report elements.

- [marginWidth](#)
- [marginRightWidth](#)
- [marginBottomWidth](#)
- [marginLeftWidth](#)
- [marginTopWidth](#)
- [borderWidth](#)
- [borderRightWidth](#)
- [borderBottomWidth](#)
- [borderLeftWidth](#)
- [borderTopWidth](#)
- [borderStyle](#)
- [borderRightStyle](#)
- [borderBottomStyle](#)
- [borderLeftStyle](#)
- [borderTopStyle](#)
- [borderColor](#)
- [borderRightColor](#)
- [borderBottomColor](#)
- [borderLeftColor](#)
- [borderTopColor](#)
- [roundedCorners](#)
- [paddingWidth](#)
- [paddingRightWidth](#)
- [paddingBottomWidth](#)
- [paddingLeftWidth](#)
- [paddingTopWidth](#)

[marginWidth](#)

Sets the thickness of all an object's margins; can be overridden by specific Margin properties. See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

[marginRightWidth](#)

Sets the thickness of an object's right margin; overrides the property [marginWidth](#). See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

[marginBottomWidth](#)

Sets the thickness of an object's bottom margin; overrides the property [marginWidth](#). See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

[marginLeftWidth](#)

Sets the thickness of an object's left margin; overrides the property [marginWidth](#). See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

marginTopWidth

Sets the thickness of an object's top margin; overrides the property marginWidth. See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

borderWidth

Sets the thickness of an object's borders; can be overridden by specific border properties. See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value. Default value is **2**.

borderRightWidth

Sets the thickness of an object's right border; overrides the borderWidth property. See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

borderBottomWidth

Sets the thickness of an object's bottom border; overrides the borderWidth property. See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

borderLeftWidth

Sets the thickness of an object's left border; overrides the borderWidth property. See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

borderTopWidth

Sets the thickness of an object's top border; overrides the borderWidth property. See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

borderStyle

Sets the style for all an object's borders. Can be overridden by specific borderSyle properties. See [Design HowTo](#) for illustrations.

Type: [Enum](#), choices are: none, solid, dotted, dashed, groove, ridge, inset, outset, double. Default is **none**.

borderRightStyle

Sets the style for an object's right border. Can override the borderStyle property. See [Design HowTo](#) for illustrations.

Type: [Enum](#), choices are: solid, dotted, dashed, groove, ridge, inset, outset, double.

borderBottomStyle

Sets the style for an object's bottom border. Can override the borderStyle property. See [Design HowTo](#) for illustrations.

Type: [Enum](#), choices are: solid, dotted, dashed, groove, ridge, inset, outset, double.

borderLeftStyle

Sets the style for an object's left border. Can override the borderStyle property. See [Design HowTo](#) for illustrations.

Type: [Enum](#), choices are: solid, dotted, dashed, groove, ridge, inset, outset, double.

borderTopStyle

Sets the style for an object's top border. Can override the `borderStyle` property. See [Design HowTo](#) for illustrations.

Type: **Enum**, choices are: solid, dotted, dashed, groove, ridge, inset, outset, double.

borderColor

The `borderColor` property sets the color of all an object's borders.

In the properties window, this property is identified by the label **Border Color**.

Can be overridden by specific `borderColor` properties. See [Design HowTo](#) for illustrations.

Type: **Color**, valid colors are selected from the Edit Expression color palette.

borderRightColor

Sets the color of an object's right border. Can override the `borderColor` property. See [Design HowTo](#) for illustrations.

Type: **Color**, valid colors are selected from the Edit Expression color palette.

borderBottomColor

Sets the color of an object's bottom border. Can override the `borderColor` property. See [Design HowTo](#) for illustrations.

Type: **Color**, valid colors are selected from the Edit Expression color palette.

borderLeftColor

Sets the color of an object's left border. Can override the `borderColor` property. See [Design HowTo](#) for illustrations.

Type: **Color**, valid colors are selected from the Edit Expression color palette.

borderTopColor

Sets the color of an object's top border. Can override the `borderColor` property. See [Design HowTo](#) for illustrations.

Type: **Color**, valid colors are selected from the Edit Expression color palette.

roundedCorners

Specifies that the object's border corners will be round. This applies to the border styles **solid**, **dashed**, and **double** only. See [Design HowTo](#) for illustrations.

Type: **Boolean**. Valid choices are True, False. The default value is False.

paddingWidth

Sets the width of all of an object's padding. Can be overridden by specific padding properties. See [Design HowTo](#) for illustrations.

Type: **PXML**, point value.

paddingRightWidth

Sets the width of an object's right padding. Can override the `paddingWidth` property. See [Design HowTo](#) for illustrations.

Type: **PXML**, point value.

paddingBottomWidth

Sets the width of an object's bottom padding. Can override the `paddingWidth` property. See [Design HowTo](#) for illustrations.

Type: **PXML**, point value.

paddingLeftWidth

Sets the width of an object's left padding. Can override the paddingWidth property. See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

paddingTopWidth

Sets the width of an object's top padding. Can override the paddingWidth property. See [Design HowTo](#) for illustrations.

Type: [PXML](#), point value.

Properties for Report Document Metadata

Report Metadata properties can be set in the report design document (.4rpt).

Report Metadata properties can be set in the 4rpt report design document. For compatibility reports, which have no 4rpt document, API functions are provided. See [Adding Report Document Metadata](#).

The metadata is inserted into the final document (such as SVG or PDF), if the format supports metadata.

- [title](#)
- [author](#)
- [creator](#)
- [subject](#)
- [keywords](#)

title

Specifies the title of the object.

Type: [String](#).

author

Specifies the metadata for the author of the report.

Type: [String](#).

creator

Specifies the metadata for the creator of the report.

Type: [String](#).

subject

Specifies the metadata for the subject of the report.

Type: [String](#).

keywords

Specifies the metadata for the keyword of the report.

Type: [String](#).

Bar Codes

The report element container for a Bar Code is a Barcode Box. This flow object draws bar codes.

These properties are specific to the Bar Code Box:

- [codeType](#)
- [fidelity](#)
- [noText](#)
- [codeValue](#)
- [check](#)
- [noDigits](#)
- [noCheckDigits](#)

- [thinToThickRelation](#)
- [thinToGapRelation](#)
- [controlCharacters](#)
- [CodeValueExpression](#)

Additional properties are inherited from the [Layout Node](#).

Properties that are specific to a bar code type are listed in the bar code type description.

Bar Code type listing

Table 181: Bar Code Types

Bar Code Type	Number of Digits Supported	Normal size
codabar-18 on page 767	varies	(calculated width x 20mm h)
codabar-2 on page 766	varies	(calculated width x 20mm h)
code-11-matrix on page 768	varies	(calculated width x 1in h)
code-128 on page 769	varies	(calculated width x 6.5mm h)
code-2-5-datalogic on page 774	varies	(calculated width x 1in h)
code-2-5-IATA on page 774	varies	(calculated width x 1in h)
code-2-5-industrial on page 774	varies	(calculated width x 1in h)
code-2-5-interleaved on page 774	varies	(calculated width x 1in h)
code-2-5-inverted on page 775	varies	(calculated width x 1in h)
code-2-5-matrix on page 775	varies	(calculated width x 1in h)
code-BCD-matrix on page 775	varies	(calculated width x 1in h)
code-32 on page 775	9	(calculated w x h)
code-39 on page 777	varies	(calculated width x 1in h)
code-39-extended on page 782	varies	(calculated width x 1in h)
code-93 on page 786	varies	(calculated w x h)
code-93-extended on page 788	varies	(calculated w X h)
data-matrix on page 792	varies	
ean-8 on page 793	8	26.73mm x 21.64 mm (w x h)
ean-13 on page 793	13	37.29mm x 26.26mm (w x h)
ean-code-128 on page 794	varies	(calculated width x 6.5mm h)
ean-data-matrix on page 794	varies	
ean-supplemental-2 on page 794	2	6.6mm x 26.26mm (w x h)
ean-supplemental-5 on page 794	5	15.5mm x 26.26mm (w x h)
gs1-8 on page 794	8	26.73mm x 21.64 mm (w x h)
gs1-13 on page 794	13	37.29mm x 26.26mm (w x h)

Bar Code Type	Number of Digits Supported	Normal size
gs1-code-128 on page 794	varies	(calculated width x 6.5mm h)
gs1-data-matrix on page 794	varies	
gs1-supplemental-2 on page 795	2	6.6mm x 26.26mm (w x h)
gs1-supplemental-5 on page 795	5	15.5mm x 26.26mm (w x h)
intelligent-mail on page 795	varies, up to 31 digits.	
pdf-417 on page 795	varies	(calculated w x h)
qr-code on page 796	varies	
upc-a on page 799	12	1.469in x 1.020in (w x h)
upc-e on page 799	8	0.897in x 1.020in (w x h)
upc-supplemental-2 on page 800	2	0.26in x 1.02in (w x h)
upc-supplemental-5 on page 800	5	(0.611in x 1.02in w x h)

Bar Code type details

A description of each code type, the expected value type, semantic constraints and size information.

The character set and semantic constraints depend on the [code type](#) selected.

codabar-2

Details on the codabar-2 bar code type.

Codabar 2 can be used to encode text of variable length by using characters from this character set:

Table 182: Character set for Codabar 2

Reference Number	Character
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	-
11	\$
12	:

Reference Number	Character
13	/
14	.
15	+
16	a
17	b
18	c
19	d

The first and the last character of any code must be either 'a', 'b', 'c' or 'd'. All other characters must have an ordinal value less than 16.

The last but one character is the checksum character that is calculated as follows: $CS=16-(\text{Sum}(i=1 \text{ to } n \text{ of Ref}(i))) \text{ mod } 16)$ where $\text{Ref}(i)$ is the reference number of the character i , and n is the total number of characters. Example: `codeValue="a37859b"` $CS=16-((16+3+7+8+5+9+17) \text{ mod } 16)$, $CS=16-(65 \text{ mod } 16)$, $CS=15$ Looking up reference number 15 yields the character '+'. The full code value including checksum is therefore: `codeValue="a37859+b"`

When the system is supplied with a value that is one digit shorter than specified by `noDigits` then the check digit is calculated automatically.

The nominal height is 20mm. The nominal width of a thin bar is $\text{THINBARWIDTH}=0.165\text{mm}$. The width of a thick bar is $\text{THICKBARWIDTH}=\text{THINBARWIDTH}/\text{thinToThickRelation}$ where `thinToThickRelation` should take values between 1/3 and 1/2. Between digits a gap having the width $\text{GAPWIDTH}=\text{THINBARWIDTH}/\text{thinToGapRelation}$ is drawn. The padding on both sides measures $10*\text{THINBARWIDTH}$.

The "American Blood Commission" defines a code type known as Codabar-ABC. There are two types of Codabar-ABC codes that can both be built using the Codabar 2 bar code type:

Single bar Codabar-ABC - this type is identical with Codabar 2, the constraint being that values have to be at least 5 digits long.

Dual bar Codabar-ABC - This type can be printed by printing two Codebar 2 horizontally adjacent to each other. The gap between the two bars should not exceed 15mm. Additionally the code of the first bar must end with a 'd' character while the second must start with the character 'd'. This code creates a valid Dual bar Codabar-ABC bar for the values "c1234d" and "d5678a".

codabar-18

Details on the codabar-18 bar code type.

Codabar 18 can be used to encode text of variable length by using characters from this character set:

Table 183: Character set for Codabar 18

Reference Number	Character
0	0
1	1
2	2
3	3
4	4
5	5

Reference Number	Character
6	6
7	7
8	8
9	9
10	-
11	\$
12	:
13	/
14	.
15	+
16	a
17	b
18	c
19	d
16	t
17	n
18	*
19	e

The first character of any code must be either 'a', 'b', 'c' or 'd'. The last character of any code must be either 't', 'n', '*' or 'e'. All other characters must have an ordinal value less than 16.

The last but one character is the checksum character that is calculated as follows: $CS = 16 - (\text{Sum}(i=1 \text{ to } n \text{ of Ref}(i))) \bmod 16$ Where Ref(i) is the reference number of the character i, and n is the total number of characters. Example: codeValue="a37859n" $CS = 16 - ((16+3+7+8+5+9+17) \bmod 16)$, $CS = 16 - (65 \bmod 16)$, $CS = 15$ Looking up reference number 15 yields the character '+'. The full code value including checksum is therefore: codeValue="a37859+n"

When the system is supplied with a value that is one digit shorter than specified by [noDigits](#) then the check digit is automatically calculated.

The nominal height is 20mm. The nominal width of a thin bar is THINBARWIDTH=0.165mm. The width of a character is 2.095mm. Between digits a gap with the width GAPWIDTH=THINBARWIDTH/[thinToGapRelation](#) is drawn. The padding on both sides measures 10*THINBARWIDTH.

code-11-matrix

Details on the code-11-matrix bar code type.

The code represents a character string with a variable number of characters. The string can contain the digits 0-9 and the '-' character. The number of digits can be specified by setting the [noDigits](#) attribute. The code can contain up to two check characters. The attribute [noCheckDigits](#) specifies how many check characters are used. If two check characters are used, the rightmost character is the 'K' checksum character and the last but one character is the 'C' checksum character. If only one checksum character is specified then the rightmost character is a 'C' checksum character. The 'C' checksum is calculated as $C = (\text{Sum}(i=1 \text{ to } n \text{ of } ((i-1 \bmod 10)+1) * \text{Ref}(n-i+1))) \bmod 11$ and the 'K' checksum is calculated using $K = (\text{Sum}(i=1 \text{ to } n \text{ of } ((i-1 \bmod 9)+1) * \text{Ref}(n-i+1))) \bmod 11$ where n specifies the number of characters to the left of the particular check digit and Ref(i) specifies

the value of the character at position i , starting with the leftmost character having the value 1. For digits $\text{Ref}()$ yields the digit value itself and for the '-' character $\text{Ref}()$ yields the value 10. Example calculating the 'C' checksum: $\text{codeValue}="12-12345-67890"$, $\text{noDigits}="16"$, $\text{noCheckDigits}="2"$, $n=14C=(1*0+2*9+3*8+4*7+\dots+10*2+1*1+2*10+3*2+4*1) \bmod 11=305 \bmod 11=8$ The K checksum can then be calculated as: $n=15$, $\text{Ref}(15)=C=8K=(1*8+2*0+3*9+4*8+\dots+9*4+1*3+2*2+3*1+4*10+5*2+6*1) \bmod 11=350 \bmod 11=9$ resulting in the code value $\text{codeValue}="12-12345-6789089"$.

If the value supplied in codeValue has the length $\text{noDigits} - \text{noCheckDigits}$ then the system automatically calculates and supplies the check digits.

The nominal height is 1 in. Digits can differ in width so that two different values having the same number of digits can result in bar codes of differing width. The nominal width of a thin bar is $\text{THINBARWIDTH}=0.0236\text{in}$. The width of a thick bar is $\text{THICKBARWIDTH}=\text{THINBARWIDTH}/\text{thinToThickRelation}$ where $\text{thinToThickRelation}$ should take values between $1/3$ and $1/2$. Between digits a gap of width $\text{GAPWIDTH}=\text{THINBARWIDTH}/\text{thinToGapRelation}$ is drawn. The default relation value is 1. The padding on both sides measures $10*\text{THINBARWIDTH}$.

code-128

Details on the code-128 bar code type.

Code 128 can be used to encode ASCII text of variable length. For this purpose characters can be selected from three character sets, each containing 106 characters. [Table 184: Available characters in the character sets A, B, and C](#) on page 769 lists the available characters in the character sets A, B and C.

Table 184: Available characters in the character sets A, B, and C

Reference Number	Character Set A	Character Set B	Character Set C
0	SP	SP	00
1	!	!	01
2	"	"	02
3	#	#	03
4	\$	\$	04
5	%	%	05
6	&	&	06
7	'	'	07
8	((08
9))	09
10	*	*	10
11	+	+	11
12	,	,	12
13	-	-	13
14	.	.	14
15	/	/	15
16	0	0	16
17	1	1	17
18	2	2	18

Reference Number	Character Set A	Character Set B	Character Set C
19	3	3	19
20	4	4	20
21	5	5	21
22	6	6	22
23	7	7	23
24	8	8	24
25	9	9	25
26	:	:	26
27	;	;	27
28	<	<	28
29	=	=	29
30	>	>	30
31	?	?	31
32	@	@	32
33	A	A	33
34	B	B	34
35	C	C	35
36	D	D	36
37	E	E	37
38	F	F	38
39	G	G	39
40	H	H	40
41	I	I	41
42	J	J	42
43	K	K	43
44	L	L	44
45	M	M	45
46	N	N	46
47	O	O	47
48	P	P	48
49	Q	Q	49
50	R	R	50
51	S	S	51
52	T	T	52

Reference Number	Character Set A	Character Set B	Character Set C
53	U	U	53
54	V	V	54
55	W	W	55
56	X	X	56
57	Y	Y	57
58	Z	Z	58
59	[[59
60	\	\	60
61]]	61
62	^	^	62
63	_	_	63
64	NUL	`	64
65	SOH	a	65
66	STX	b	66
67	ETX	c	67
68	EOT	d	68
69	ENQ	e	69
70	ACK	f	70
71	BEL	g	71
72	BS	h	72
73	HT	i	73
74	LF	j	74
75	VT	k	75
76	FF	l	76
77	CR	m	77
78	SO	n	78
79	SI	o	79
80	DLE	p	80
81	DC1	q	81
82	DC2	r	82
83	DC3	s	83
84	DC4	t	84
85	NAK	u	85
86	SYN	v	86

Reference Number	Character Set A	Character Set B	Character Set C
87	ETB	w	87
88	CAN	x	88
89	EM	y	89
90	SUB	z	90
91	ESC	{	91
92	FS		92
93	GS	}	93
94	RS	~	94
95	US	DEL	95
96	FNC3	FNC3	96
97	FNC2	FNC2	97
98	SHIFT	SHIFT	98
99	CODEC	CODEC	99
100	CODEB	FNC4	CODEB
101	FNC4	CODEA	CODEA
102	FNC1	FNC1	FNC1
103	STARTA	STARTA	STARTA
104	STARTB	STARTB	STARTB
105	STARTC	STARTC	STARTC
-	STOP	STOP	STOP

The code value is expected as a comma-separated list of character names. It must start with a character set selection character STARTA, STARTB, or STARTC and must end with a checksum character followed by the STOP character. If these characters are omitted then the system calculates the checksum automatically and adds the required STOP character.

The control characters CODEA, CODEB and CODEC can be used to switch from one character set to another.

The control character SHIFT changes the character set for the immediately following character from A to B and vice versa.

The smartParse property

The [smartParse](#) property can be used when the code value consists solely of printable characters. This alleviates users of the need to manually select character sets. When enabled, the resulting bar code is encoded with a shortest possible encoding, for the given string, producing a minimally sized visual representation.

EAN 128 (GS1 128) bar codes

EAN 128 bar codes can be drawn using this bar code type.

Note: [ean-code-128](#) on page 794 and [gs1-code-128](#) on page 794 bar codes are synonymous.

Valid EAN 128 codes start with the sequence "STARTC,FNC1".

Tip: With ean-code-128 or gs1-code-128, the data passes as a string (using smart parse) and the "STARTC,FNC1," is added by the engine. If the ean-code-128 or gs1-code-128 were not made available, you would have to manually encode the string starting with "STARTC,FNC1," and then manually select the appropriate character sets to encode the data.

What follows is a sequence of data packets. Each packet starts with a one digit application identifier (AI) from the C character set. The AI is followed by data. The type and amount of expected data is AI specific. The amount can be fixed or variable. In the case of variable amount of data, the end of the data must be indicated by a FNC1 character. Here is a table with some common AIs.

Table 185: EAN 128 Bar Code Common AIs

AI	No of Data Characters	Description
00	18	Identification of a delivery unit
01	14	An EAN 13 Number including check digit
10	up to 20 alphanumeric characters	Shipping batch identifier
11	3	Production date in the format YYMMDD

The complete list of AIs is country specific and is maintained by the local EAN organization. The textual representation requires AIs to be enclosed in round braces. Examples:

Table 186: EAN 128 Bar Code textual representation

Textual representation	code value	Remark
(25)03x57	STARTC, FNC1, 25, 03, CODEB, x, 5, 7	Note that although, it is a variable length AI, no FNC1 is added at the end in the case when the AI is the last one.
(30)19(21)3456789012	STARTC, FNC1, 30, 19, CODEA, FNC1, CODEC, 21, 12, 34, 56, 78, 90, 12	Note that switching to code a is not necessary, as FNC1 exists in all three character sets.

If a code contains the control character FNC2 then the decoder does not transmit this value. Instead it appends the value to an internal storage. Only after the decoder encounters a value not containing the FNC2 control character, the decoder transmits the temporary storage and the value read. The temporary storage is then cleared. The purpose of this mechanism is to allow the breaking of long text sequences into several lines.

The effects of the control characters FNC3 and FNC4 are decoder specific.

Decoders remove the character STARTA, STARTB, STARTC, CODEA, CODEB, CODEC, SHIFT, FNC1, FNC2, FNC3, FNC4, STOP, and the checksum character from the data before displaying or transmitting the value.

The last but one character is the checksum character that is calculated as follows: $CS = (\text{Ref}(1) + \sum_{i=2}^n ((i-1) * \text{Ref}(i))) \bmod 103$ where $\text{Ref}(i)$ is the reference number of the character i , and n is the total number of characters. Example: $\text{codeValue} = \text{"STARTB,A,B,C"}$ $CS = (104 + (1 * 33) + (2 * 34) + (3 * 35)) \bmod 103$, $CS = 310 \bmod 103$, $CS = 1$ Looking up reference number 1 in character set B yields the exclamation mark '!' character. The full code value including checksum and stop character is therefore: $\text{codeValue} = \text{"STARTB,A,B,C,!,STOP"}$

Height and width

The nominal height is 6.5mm. The width of the bar can be calculated using this formula: $L/mm=(5.5N_c + 11N_{ab} + 35) * 0.19$ where N_c is the number of characters from character set C and N_{ab} denotes the number of characters from character sets A and B. On each side of the bar area 1.9mm padding is added.

code-2-5-datalogic

Details on the code-2-5-datalogic bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the [noDigits](#) attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by [noDigits](#) then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars making a total of 5 bars per digit. The nominal width of a thin bar is $THINBARWIDTH=0.0236in$. The width of a thick bar is $THICKBARWIDTH=THINBARWIDTH/$ [thinToThickRelation](#) where [thinToThickRelation](#) [thinToGapRelation](#) is drawn. The padding on both sides measures $10 * THINBARWIDTH$.

code-2-5-IATA

Details on the code-2-5-IATA bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the [noDigits](#) attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by [noDigits](#) then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars yielding a total of 5 bars per digit. The nominal width of a thin bar is $THINBARWIDTH=0.0236in$. The width of a thick bar is $THICKBARWIDTH=THINBARWIDTH/$ [thinToThickRelation](#) where [thinToThickRelation](#) should take values between $1/3$ and $1/2$. Between digits a gap of width $GAPWIDTH=THINBARWIDTH/$ [thinToGapRelation](#) is drawn. The padding on both sides measures $10 * THINBARWIDTH$.

code-2-5-industrial

Details on the code-2-5-industrial bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the [noDigits](#) attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by [noDigits](#) then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars yielding a total of 5 bars per digit. The nominal width of a thin bar is $THINBARWIDTH=0.0236in$. The width of a thick bar is $THICKBARWIDTH=THINBARWIDTH/$ [thinToThickRelation](#) where [thinToThickRelation](#) should take values between $1/3$ and $1/2$. Between digits a gap of the width $GAPWIDTH=THINBARWIDTH/$ [thinToGapRelation](#) ([Thin To Gap Relation](#)) on page 757 is drawn. The padding on both sides measures $10 * THINBARWIDTH$.

code-2-5-interleaved

Details on the code-2-5-interleaved bar code type.

The code represents a decimal number with a variable number of digits. The number of digits must be a multiple of 2. The number of digits can be specified by setting the [noDigits](#) attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by [noDigits](#) then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars resulting in a total of 5 bars per digit. The nominal width of a thin bar is $THINBARWIDTH=0.0236in$. The width of a thick bar is $THICKBARWIDTH=THINBARWIDTH/ thinToThickRelation$ where $thinToThickRelation$ should take values between $1/3$ and $1/2$. Between digits a gap of width $GAPWIDTH=THINBARWIDTH/ thinToGapRelation$ is drawn. The padding on both sides measures $10*THINBARWIDTH$.

code-2-5-inverted

Details on the code-2-5-inverted bar code type.

The code is the same as [code 2/5 industrial](#), the only difference being that the gaps are drawn instead of the bars.

code-2-5-matrix

Details on the code-2-5-matrix bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the [noDigits](#) attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by [noDigits](#) then the check digit is automatically calculated by the system.

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars yielding a total of 5 bars per digit. The nominal width of a thin bar is $THINBARWIDTH=0.0236in$. The width of a thick bar is $THICKBARWIDTH=THINBARWIDTH/ thinToThickRelation$ where $thinToThickRelation$ should take values between $1/3$ and $1/2$. Between digits a gap of width $GAPWIDTH=THINBARWIDTH/ thinToGapRelation$ is drawn. The padding on both sides measures $10*THINBARWIDTH$.

code-BCD-matrix

Details on the code-BCD-matrix bar code type.

The code represents a decimal number with a variable number of digits. The number of digits can be specified by setting the [noDigits](#) attribute. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a value that is one digit shorter than specified by [noDigits](#) then the check digit is automatically calculated by the system.

The nominal height is 1in. Digits can differ in width so that two different values having the same number of digits can result in bar codes of differing width. The nominal width of a thin bar is $THINBARWIDTH=0.0236in$. The width of a thick bar is $THICKBARWIDTH=THINBARWIDTH/ thinToThickRelation$ where $thinToThickRelation$ should take values between $1/3$ and $1/2$. Between digits a gap of width $GAPWIDTH=THINBARWIDTH/ thinToGapRelation$ is drawn. The default relation value is 1. The padding on both sides measures $10*THINBARWIDTH$.

code-32

Details on the code-32 bar code type.

The code represents a decimal number with 9 digits. The last digit is the check character. The check character is calculated as follows: $CS=Sum(i=1 \text{ to } 8 \text{ of } CT((i-1)*2*Ref(i))) \text{ mod } 10$ where $Ref(i)$ denotes the value of the digit at position i (the leftmost digit has the index 1) and $CT()$ denotes the cross total of its argument (e.g. $CT(18)=1+8=9$).

The rightmost digit (checksum digit) can be omitted. When the system is supplied with a 8 digit value then the check digit is automatically calculated.

The specified 9 digit decimal value is translated into a 6 digit base 32 value which is then drawn using characters from the base 39 character set and bar drawing scheme. This character table is used to encode the 6 digit base 32 value:

Table 187: 32 character table is used to encode the 6 digit base 32 value

Reference Number	Character
0	0
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	B
10	C
11	C
12	D
13	F
14	G
15	H
16	J
17	K
18	L
19	M
20	N
21	P
22	Q
23	R
24	S
25	T
26	U
27	V
28	W
29	X
30	Y
31	Z

Based on this table the following bar codes will produce identical bar pattern: <BARCODEBOX codeType="code 32" check="true" codeValue="026089019" noText="true" orientation="vertical" mirrored="true"/><BARCODEBOX codeType="code 39" check="false" codeValue="0SW5KV" noText="true" orientation="vertical" mirrored="true"/> Note that the bars are drawn without text by setting noText="true". This is necessary to produce identical output since otherwise the code 32 box draws the decimal number "026089019" while the code 39 box would draw the string "0SW5KV".

The nominal height is 1in. Each digit is drawn using a pattern of two wide and three thin bars making a total of 5 bars per digit. The nominal width of a thin bar is THINBARWIDTH=0.0197in. The width of a thick bar is THICKBARWIDTH=THINBARWIDTH/ [thinToThickRelation](#) where [thinToThickRelation](#) should take values between 1/3 and 1/2. Between digits a gap with the width GAPWIDTH=THINBARWIDTH/ [thinToGapRelation](#) is drawn. The padding on both sides measures 10*THINBARWIDTH.

code-39

Details on the code-39 bar code type.

Code 39 can be used to encode ASCII text of variable length. Characters can be selected from this set of characters.

Table 188: Characters for encoding ASCII text of variable length for Code 39

Reference Number	Character
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	G
17	H
18	I
19	J
20	K
21	L

Reference Number	Character
22	M
23	N
24	O
25	P
26	Q
27	R
28	S
29	T
30	U
31	V
32	W
33	X
34	Y
35	Z
36	-
37	.
38	
39	\$
40	/
41	+
42	%

The last but one character is the checksum character that is calculated as follows: $CS = \text{Sum}(i=1 \text{ to } n \text{ of Ref}(i)) \text{ mod } 43$ where $\text{Ref}(i)$ is the reference number of the character i , and n is the total number of characters. Example: `codeValue="DATALOGIC"` $CS = (13+10+29+10+21+24+16+18+12) \text{ mod } 43, CS=153 \text{ mod } 43, CS=24$ Looking up reference number 24 yields the character 'O'. The full code value including checksum and stop character is therefore: `codeValue="DATALOGICO"`

Some scanners support an "extended mode" in which the scanner recognizes special two-character sequences of the code 39 character set and decodes these as ASCII characters. With this method the full 128 character ASCII character set can be encoded using the 43 basic characters of code 39. Scanners are switched into "extended mode" by a bar containing the sequence "+\$". The sequence "-\$" returns the scanner into regular mode. This table lists the character sequences needed to encode the ASCII character set in "extended mode".

Table 189: Character sequences needed to encode the ASCII character set in "extended mode"

ASCII code	Code 39 sequence
NUL	%U
SOH	\$A
STX	\$B

ASCII code	Code 39 sequence
ETX	\$C
EOT	\$D
ENQ	\$E
ACK	\$F
BEL	\$G
BS	\$H
HT	\$I
LF	\$J
VT	\$K
FF	\$L
CR	\$M
SO	\$N
SI	\$O
DLE	\$P
DC1	\$Q
DC2	\$R
DC3	\$S
DC4	\$T
NAK	\$U
SYN	\$V
ETB	\$W
CAN	\$X
EM	\$Y
SUB	\$Z
ESC	%A
FS	%B
GS	%C
RS	%D
US	%E
SP	
!	/A
"	/B
#	/C
\$	/D

ASCII code	Code 39 sequence
%	/E
&	/F
'	/G
(/H
)	/I
*	/J
+	/K
,	/L
-	-
.	.
/	/O
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
:	/Z
;	%F
>TD/TD	
=	%H
>	%I
?	%J
@	%V
A	A
B	B
C	C
D	D
E	E
F	F

ASCII code	Code 39 sequence
G	G
H	H
I	I
J	J
K	K
L	L
M	M
N	N
O	O
P	P
Q	Q
R	R
S	S
T	T
U	U
V	V
W	W
X	X
Y	Y
Z	Z
[%K
\	%L
]	%M
^	%N
_	%O
`	%W
a	+A
b	+B
c	+C
d	+D
e	+E
f	+F
g	+G
h	+H

ASCII code	Code 39 sequence
i	+I
j	+J
k	+K
l	+L
m	+M
n	+N
o	+O
p	+P
q	+Q
r	+R
s	+S
t	+T
u	+U
v	+V
w	+W
x	+X
y	+Y
z	+Z
{	%P
	%Q
}	%R
~	%S
DEL	%T

The code type "code 39 extended" automatically creates the necessary two-character sequences and can be used as a more convenient way of creating extended code 39 bar codes.

If the value specified by `codeValue` is shorter by one character than the number of digits specified by `noDigits` then the checksum character is automatically calculated and the character is appended to `codeValue`.

The nominal height is 1in. The nominal width of a thin bar is $\text{THINBARWIDTH}=0.0197\text{in}$. The width of a thick bar is $\text{THICKBARWIDTH}=\text{THINBARWIDTH}/\text{thinToThickRelation}$ where `thinToThickRelation` should take values between 1/3 and 1/2. Between digits a gap with the width $\text{GAPWIDTH}=\text{THINBARWIDTH}/\text{thinToGapRelation}$ is drawn. The default relation value is 1. The padding on both sides measures $10*\text{THINBARWIDTH}$.

code-39-extended

Details on the code-39-extended bar code type.

Code 39 Extended can be used to encode text of variable length using the ASCII character set. Depending on the value of the `smartParse` property, the code value is either expected as a comma-separated list of ASCII character names or as a plain string. [Table 190: Code 39 Extended names and the textual](#)

[representation that is used in printout](#) on page 783 lists the names and the textual representation that is used in printout.

Table 190: Code 39 Extended names and the textual representation that is used in printout

ASCII code	Textual representation
NUL	<NUL>
SOH	<SOH>
STX	<STX>
ETX	<ETX>
EOT	<EOT>
ENQ	<ENQ>
ACK	<ACK>
BEL	<BEL>
BS	<BS>
HT	<HT>
LF	<LF>
VT	<VT>
FF	<FF>
CR	<CR>
SO	<SO>
SI	<SI>
DLE	<DLE>
DC1	<DC1>
DC2	<DC2>
DC3	<DC3>
DC4	<DC4>
NAK	<NAK>
SYN	<SYN>
ETB	<ETB>
CAN	<CAN>
EM	
SUB	<SUB>
ESC	<ESC>
FS	<FS>
GS	<GS>
RS	<RS>

ASCII code	Textual representation
US	<US>
SP	
!	!
"	"
#	#
\$	\$
%	%
&	&
'	'
((
))
*	*
+	+
COMMA	,
-	-
.	.
/	/
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
:	:
;	;
<	<
=	=
>	>
?	?
@	@

ASCII code	Textual representation
A	A
B	B
C	C
D	D
E	E
F	F
G	G
H	H
I	I
J	J
K	K
L	L
M	M
N	N
O	O
P	P
Q	Q
R	R
S	S
T	T
U	U
V	V
W	W
X	X
Y	Y
Z	Z
[[
\	\
]]
^	^
_	_
`	`
a	a
b	b

ASCII code	Textual representation
c	c
d	d
e	e
f	f
g	g
h	h
i	i
j	j
k	k
l	l
m	m
n	n
o	o
p	p
q	q
r	r
s	s
t	t
u	u
v	v
w	w
x	x
y	y
z	z
{	{
}	}
~	~
DEL	

code-93

Details on the code-93 bar code type.

Code 93 can be used to encode ASCII text of variable length. Characters can be selected from this set of characters. You can use the [code-93-extended](#) type to encode the full 128 character ASCII character set using the 47 basic characters of code 93.

Table 191: code-93

Reference Number	Character
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	G
17	H
18	I
19	J
20	K
21	L
22	M
23	N
24	O
25	P
26	Q
27	R
28	S
29	T
30	U
31	V
32	W

Reference Number	Character
33	X
34	Y
35	Z
36	-
37	.
38	
39	\$
40	/
41	+
42	%
43	!
44	?
45	\
46	

The textual representation of the last four characters "!?|\|" can be configured by setting the [controlCharacters](#) attribute.

[noCheckDigits](#) specifies how many check characters are used. In the case of two check characters the rightmost character is the 'K' checksum character and the last but one character is the 'C' checksum character. If only one checksum character is specified then the rightmost character is a 'C' checksum character. The 'C' checksum is calculated as follows $C = (\text{Sum}(i=1 \text{ to } n \text{ of } ((i-1 \text{ mod } 20)+1) * \text{Ref}(n-i+1))) \text{ mod } 47$ and the 'K' checksum is calculated as follows $K = (\text{Sum}(i=1 \text{ to } n \text{ of } ((i-1 \text{ mod } 15)+1) * \text{Ref}(n-i+1))) \text{ mod } 47$ where n specifies the number of characters left of the particular check digit and $\text{Ref}(i)$ specifies the reference value of the character at position i starting with the leftmost character having the index value 1. Reference numbers can be looked up in the first column. Example calculating the 'C' checksum: `codeValue="DATALOGIC"`, `noDigits="11"`, `noCheckDigits="2"`, $n=9$ $C = (1*12+2*18+3*16+4*24+\dots+7*29+8*10+9*13) \text{ mod } 47 = 757 \text{ mod } 47 = 5$ The K checksum can then be calculated as: $n=10$, $\text{Ref}(10)=C=5$ $K = (1*5+2*12+3*18+4*16+\dots+8*29+9*10+10*13) \text{ mod } 47 = 915 \text{ mod } 47 = 22$ resulting in the code value `codeValue="DATALOGIC5M"`.

If the value supplied in `codeValue` has the length `noDigits- noCheckDigits` then the system automatically calculates and supplies the check digits.

code-93-extended

Details on the code-93-extended bar code type.

Some scanners support an "extended mode" in which the scanner recognizes special two-character sequences of the code 93 character set and decodes these as ASCII characters. With this mode, the full 128 character ASCII character set can be encoded using the 47 basic characters of code 93. T

The nominal height is 1in. The nominal width of a bar is $\text{BARWIDTH}=0.022\text{in}$. The width of the entire bar is $(1+(\text{noDigits}+2)*9)*\text{BARWIDTH}$. The padding on both sides measures $10*\text{BARWIDTH}$.

This code type automatically creates the necessary two-character sequences and can be used as a more convenient way of creating extended code 93 bar codes.

Table 192: code-93-extended sequence table

ASCII code	Code 93 sequence
NUL	?U
SOH	!A
STX	!B
ETX	!C
EOT	!D
ENQ	!E
ACK	!F
BEL	!G
BS	!H
HT	!I
LF	!J
VT	!K
FF	!L
CR	!M
SO	!N
SI	!O
DLE	!P
DC1	!Q
DC2	!R
DC3	!S
DC4	!T
NAK	!U
SYN	!V
ETB	!W
CAN	!X
EM	!Y
SUB	!Z
ESC	?A
FS	?B
GS	?C
RS	?D
US	?E
SP	

ASCII code	Code 93 sequence
!	\A
"	\B
#	\C
\$	\$
%	%
&	\F
'	\G
(\H
)	\I
*	\J
+	+
COMMA	\L
-	-
.	.
/	/
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
:	\Z
;	?F
<	?G
=	?H
>	?I
?	?J
@	?V
A	A
B	B

ASCII code	Code 93 sequence
C	C
D	D
E	E
F	F
G	G
H	H
I	I
J	J
K	K
L	L
M	M
N	N
O	O
P	P
Q	Q
R	R
S	S
T	T
U	U
V	V
W	W
X	X
Y	Y
Z	Z
[?K
\	?L
]	?M
^	?N
_	?O
`	?W
a	A
b	B
c	C
d	D

ASCII code	Code 93 sequence
e	E
f	F
g	G
h	H
i	I
j	J
k	K
l	L
m	M
n	N
o	O
p	P
q	Q
r	R
s	S
t	T
u	U
v	V
w	W
x	X
y	Y
z	Z
{	?P
	?Q
}	?R
~	?S
DEL	?T

data-matrix

Details on the data-matrix bar code type.

A Data matrix bar code can be used to encode text and binary data of variable length. It is possible to encode up to 1558 code words. Text, binary data and numeric data are compressed into code words so that the limits for these types are:

- Alphanumeric data - up to 2335 characters
- 8-bit byte data - 1555 characters
- Numeric data - 3116 digits

The limits for mixed text are lower, as control characters are inserted to switch between the different compression modes.

Data matrix symbols exist in several flavors with varying methods and degrees of error correction. Of the possible options ECC 000, ECC 050, ECC 080, ECC 100, ECC 140 and ECC 200 this implementation supports only ECC 200 using Reed Solomon error correction.

Special control characters like FNC1, ECI and "structured append" are currently not available.

The conversion of the data specified in this attribute (codeValue) to the internal representation is done by an algorithm that minimizes space. Byte values that cannot be represented in an XML document (for example all characters lower than 0x20 except 0x9, 0xa and 0xd) can be represented by a backslash (\) character followed by a 3 digit octal literal. The backslash character itself can be escaped by a sequence of two backslash characters.

The current implementation can encode any character that exists in the code page ISO-8859-1 (Latin 1). Any attempt to encode other characters will fail. Future versions will insert ECI control characters to switch to other code pages if the character is available there.

This attribute is unique to this bar code type:

- preferRectangularSymbols
 - Type: Boolean value
 - Description: If you are concerned about running out of space in the vertical of the page, you might prefer a symbol that is wider than it is high. This property produces a rectangular shaped symbol if the encoded data does not exceed 49 code words.
 - Fails if: Value cannot be parsed as a boolean value. Valid values are True, False.
 - Default value: False

ean-8

Details on the ean-8 bar code type.

Note: The ean-8 and [gs1-8](#) on page 794 bar codes are synonymous. The gs1-8 synonym can be used for the ean-8 bar code.

The code represents a 8 digit decimal number. First two digits select the country. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with such a 7 digit value then the 8th digit is automatically calculated by the system.

The nominal size is 26.73mm x 21.64mm (w x h). The bar code area has the measurements 22.11mm x 19.88mm. The left padding measures 2.31mm.

ean-13

Details on the ean-13 bar code type.

Note: The ean-13 and [gs1-13](#) on page 794 bar codes are synonymous. The gs1-13 synonym can be used for the ean-13 bar code.

The code represents a 13-digit decimal number. First two digits are the flag code, the next ten digits are the data characters and the last digit is the check character. The check character is calculated as follows:

1. Designate the rightmost character odd.
2. Sum all of the characters in the odd positions and multiply the result by three.
3. Sum all of the characters in the even positions.
4. Add the odd and even totals from steps two and three.
5. Determine the smallest number that, when added to the result from step four, will result in a multiple of 10. This is the check character.

In Europe the first two characters (flag code) are the country identifier and the data characters are split into two groups of five digits each where the first five are a company identifier and the latter five identify an article within that company. The last digit is the check character.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with such a 12 digit value then the 13th digit is automatically calculated by the system.

The nominal size is 37.29mm x 26.26mm (w x h). The bar code area has the measurements 31.35mm x 24.50mm. The left padding measures 3.63mm.

ean-code-128

Details on the ean-code-128 bar code type.

Based on the [code-128](#) on page 769 bar code.

Note: The ean-code-128 and [gs1-code-128](#) on page 794 bar codes are synonymous. The gs1-code-128 synonym can be used for the ean-code-128 bar code.

ean-data-matrix

Details on the ean-data-matrix bar code type.

Based on the [data-matrix](#) bar code. The ean-data-matrix supports smart parse, eliminating the need to manually insert the FNC1 character into the bar code.

Note: The ean-data-matrix and [gs1-data-matrix](#) on page 794 bar codes are synonymous. The gs1-data-matrix synonym can be used for the ean-data-matrix bar code.

ean-supplemental-2

Details on the ean-supplemental-2 bar code type.

The code represents a 2 digit decimal number that can be used in conjunction with the ean 13 bar code type. Typically the code is placed to the right of the ean 13 bar code. The gap between the rightmost bar of the ean 13 code and the leftmost bar of the supplemental code should not be less than 2.31mm and should not exceed 3.3mm.

The nominal size is 6.6mm x 26.26mm (w x h). The bar code is painted without padding at the sides.

Note: The ean-supplemental-2 and [gs1-supplemental-2](#) on page 795 bar codes are synonymous. The gs1-supplemental-2 synonym can be used for the ean-supplemental-2 bar code.

ean-supplemental-5

Details on the ean-supplemental-5 bar code type.

The code represents a 5 digit decimal number that can be used in conjunction with the ean 13 bar code type. Typically the code is placed to the right of the ean 13 bar code. The gap between the rightmost bar of the ean 13 code and the leftmost bar of the supplemental code should not be less than 2.31mm and should not exceed 3.3mm.

The nominal size is 15.5mm x 26.26mm (w x h). The bar code is painted without padding at the sides.

Note: The ean-supplemental-5 and [gs1-supplemental-5](#) on page 795 bar codes are synonymous. The gs1-supplemental-5 synonym can be used for the ean-supplemental-5 bar code.

gs1-8

A synonym of the ean-8 bar code type.

See [ean-8](#) on page 793.

gs1-13

A synonym of the ean-13 bar code type.

See [ean-13](#) on page 793.

gs1-code-128

A synonym for the ean-code-128 bar code type.

See [ean-code-128](#) on page 794.

gs1-data-matrix

A synonym for the ean-data-matrix bar code type.

See [ean-data-matrix](#) on page 794.

gs1-supplemental-2

A synonym of the ean-supplemental-2 bar code type.

See [ean-supplemental-2](#) on page 794.

gs1-supplemental-5

A synonym of the ean-supplemental-5 bar code type.

See [ean-supplemental-5](#) on page 794.

intelligent-mail

Details on the Intelligent Mail bar code type.

Intelligent Mail is a United States postal service bar code for usage with the USPS mail stream. It is also known as the USPS OneCode Solution or USPS 4-State Customer Barcode. Valid abbreviations for this bar code type include 4CB, 4-CB, and USPS4CB.

The bar code encodes up to 31 digits. The code value is expected as two comma-separated fields. Each field consists solely of digits. The first field contains the tracking code. The second field contains the routing code.

The encoding is illustrated in the following table. There can be a total of 31 digits maximum.

Table 193: Encoding for the Intelligent Mail bar code

Type	Field	Digits
Tracking code field	Barcode Identifier	2 digits; 2nd digit must be 0-4.
Tracking code field	Service Type Identifier	3 digits.
Tracking code field	Mailer Identifier	6 or 9 digits.
Tracking code field	Serial Number	9 digits, when used with a 6-digit Mailer Identifier. 6 digits, when used with a 9-digit Mailer Identifier.
Routing code field	Delivery Point ZIP Code	0, 5, 9, or 11 digits.

pdf-417

Details on the pdf-417 bar code type.

Pdf-417 can be used to encode text and binary data of variable length. It is possible to encode up to 925 code words (minimal error correction degree). Text, binary data and numeric data are compressed into code words so that the limits for these types are:

- 1850 ASCII characters
- 1108 Byte values
- 2710 numeric values

The limits for mixed text are lower, as control characters are inserted to switch between the different compression modes.

Setting the error correction degree to the recommended minimum lowers the maximum number of encodable code words to 863 yielding these limits for the encodable types:

- 1726 ASCII characters
- 1033 Byte values
- 2528 numeric values

The conversion of the data specified in this attribute (codeValue) to the internal representation is done by an algorithm that minimizes space (Sometimes there is more than one option to encode a particular piece of data). Non ascii characters are encoded using the specified [encoding](#) table. Byte values that cannot be represented in an XML document (for example all characters lower 0x20 except 0x9, 0xa and 0xd) can be

represented by a backslash (\) character followed by a 3 digit octal literal. The backslash character itself can be escaped by a sequence of two backslash characters.

These attributes (properties) are unique to this bar code type:

- [dataSymbolsPerLine \(Data Symbols per Line\)](#) on page 743:
 - Type: Integer value.
 - Specifies the number of data symbols per line. The value must be an integer between 1 and 30. Low values cause more narrow printout with more lines. The number of lines is not allowed to exceed 90. It should be noted, that the overall required image space usually grows with lower values because there is a constant amount of organizational information which is added with each additional line. This is not generally the case, since lines have to be filled with padding so that specially with small amounts of data a larger value may actually create a larger image. If the value is not specified, the system computes a value that minimizes image space.
 - Fails if: Value cannot be parsed as a integer value. Value is not in the range 1...30.
 - Default value: A value that minimizes the overall image size.
- [rawCodeValue \(Raw Code Value\)](#) on page 753:
 - Type: A comma-separated list of integers in the range 0...899.
 - This attribute can be used instead of [codeValue](#) to specify the code value at a lower level giving more control on the encoded data.
 - Fails if: Encoding for non-ASCII characters in the code value.
 - Default value: not set
- [errorCorrectionDegree \(Error Correction Degree\)](#) on page 746:
 - Type: Integer value.
 - Specifies the error correction degree. Valid values are in the range 0...8. Higher values make the image more robust.
 - Fails if: Value cannot be parsed as a integer value. Value is not in the range 0...8.
 - Default value: A value that proportional to the data size
- [encoding \(Encoding\)](#) on page 745:
 - Type: Encoding
 - Sets the encoding for non ascii characters in the code value. Run "`java CharsetInfo`" for a list of character set encodings available on a particular platform. Valid example values are 'ISO-8859-15' or 'IBM437'.
 - Fails if:
 - Value is not a valid host name
 - Socket connection cannot be established
 - Default value: not set (the lower 8 bits of the unicode values are encoded)

qr-code

QR code (abbreviated from Quick Response Code) is a type of matrix bar code consisting of black modules (square dots) arranged in a square grid on a white background.

The QR code can be used to encode text and binary data of variable length. It is possible to encode up to 2953 code words. The code value is converted from the XML unicode value to the specified encoding.

By default, the bytes encoded in a QR code image are interpreted as characters from the ISO-8859-1 (Latin 1) encoding. Other encodings can be specified in the encoding attribute and an *extended channel interpretations* (ECI) code will be inserted if available for the encoding. The code covers issues such as:

- Is the data compressed? If yes, how?
- Is the data part of a larger message? If yes, which part?
- Is the data encoded in a non standard way? If yes, which encoding was used?

An ECI code exists for the following encodings:

- Cp437 (0,2)
- ISO-8859-1 (ECI codes 1,3)
- ISO-8859-2 (ECI code 4)
- ISO-8859-3 (ECI code 5)
- ISO-8859-4 (ECI code 6)
- ISO-8859-5 (ECI code 7)
- ISO-8859-6 (ECI code 8)
- ISO-8859-7 (ECI code 9)
- ISO-8859-8 (ECI code 10)
- ISO-8859-9 (ECI code 11)
- ISO-8859-10 (ECI code 12)
- ISO-8859-11 (ECI code 13)
- ISO-8859-13 (ECI code 15)
- ISO-8859-14 (ECI code 16)
- ISO-8859-15 (ECI code 17)
- ISO-8859-16 (ECI code 18)
- Shift_JIS (ECI code 20)
- Cp1250, windows-1250(ECI code 21)
- Cp1251, windows-1251(ECI code 22)
- Cp1252, windows-1252(ECI code 23)
- Cp1256, windows-1256(ECI code 24)
- UnicodeBigUnmarked, UTF-16BE, UnicodeBig (ECI code 25)
- UTF-8 (ECI code 26)
- US-ASCII (ECI codes 27,170)
- Big5 (ECI code 28)
- GB18030, GB2312, EUC_CN, GBK (ECI code 29)
- EUC-KR (ECI code 30)

Attributes unique to this bar code type:

- errorCorrectionDegree
 - Type: Integer value
 - Description: Specifies the error correction degree. Valid values are in the range of 0 - 3. Higher values make the image more robust (ISO 18004:2006, 6.5.1 defines: 0=~7%, 1=~15%, 2=~25% and 3=~30%).
 - Fails if: Value cannot be parsed as an integer value. Value is not in the range of 0 - 3.
 - Default value: 3
- encoding
 - Type: encoding
 - Description: Sets the encoding for non-ASCII characters in the code value. The following encodings can be set:

Encoding Type	Description
ISO-8859-1	Use this setting if all characters in codeValue are from this code page. Some scanners or scanner apps interpret the non ASCII characters non standard (e.g. as Japanese characters). In this case the scanner may have a setting to change the interpretation. If this is not the case then try using "UTF-8" encoding. Please note that for ISO-8859-1 encoding no ECI code is

Encoding Type	Description
	embedded (Use the encoding ISO-8859-15 to force an ECI inclusion.)
Bytes	Use this setting to set the byte values. Literal characters in codeValue are mapped to ISO-8859-1 byte representation and characters not representable in XML documents can be escaped by a backslash ('\') character followed by a 3 digit octal literal. The backslash character itself can be escaped by a sequence of two backslash characters or its octal representation \134 in ISO-8859-1.
UTF-8	This is the default value and should be used unless the two other options from above are applicable. If a scanner fails to interpret the characters correctly and all character in codeValue are available from a different encoding listed by CharSetInfo, then this encoding should be tried next.
Any encoding listed by CharSetInfo (e.g. "Shift_JIS", "Big5" or "ISO-8859-8")	If all of the option above are not applicable this option should be used. For encodings for which a ECI code exists, the code will be embedded allowing the scanner to change the interpretation accordingly.
Any encoding listed by CharSetInfo prefixes with the string "RAW-" (e.g. "RAW-Shift_JIS", "RAW-Big5" or "RAW-UTF-8")	Some scanners do not recognize ECI codes and expect a specific encoding (other than ISO-8859-1). For this case, this setting should be used.

- Fails if: Value is not a valid encoding name.
- Default value: UTF-8

QR code examples



Figure 379: Hello World: size not specified (default error correction (3))



Figure 380: Hello World: size not specified (error correction degree (2))



Figure 381: Hello World: size not specified (error correction degree (1))



Figure 382: Hello World: width="3cm"



Figure 383: <http://www.4js.com>

upc-a

Details on the upc-a bar code type.

The code represents a 12 digit decimal number. First digit is the 'number system character' code, the next five identify the manufacturer and the following five identify an article. The last digit is the check character. The check character is calculated the same way as the ean 13 check sum.

The rightmost digit (checksum digit) can be omitted. When the system is supplied with such a 11 digit value then the 12th digit is automatically calculated by the system.

The nominal size is 1.469in x 1.020in (w x h). The bar code area has the measurements 1.235in x 0.965in. The left padding measures 0.117in.

upc-e

Details on the upc-e bar code type.

The code represents a 8 digit decimal number. Upc-e is a compressed version of upc-a. By the method of zero suppression some upc a codes are available as upc e codes. This translation table shows which upc-a numbers can be transformed to short versions:

Table 194: upc-a numbers that can be transformed to short versions

UPC-E Number	Digits to insert	Position of Insertion	Resulting UPC-A Number
sNNNNN0c	00000	3	sNN00000NNNc
sNNNNN1c	10000	3	sNN10000NNNc
sNNNNN2c	20000	3	sNN20000NNNc
sNNNNN3c	00000	4	sNNN00000NNc
sNNNNN4c	00000	5	sNNNN00000Nc
sNNNNN5c	00005	6	sNNNNN00005c
sNNNNN6c	00006	6	sNNNNN00006c
sNNNNN7c	00007	6	sNNNNN00007c
sNNNNN8c	00008	6	sNNNNN00008c
sNNNNN9c	00009	6	sNNNNN00009c

The rightmost digit (checksum digit) can be omitted. When the system is supplied with such a 7 digit value then the 8th digit is automatically calculated by the system.

The nominal size is 0.897in x 1.020in (w x h). The bar code area has the measurements 0.663in x 0.965in. The left padding measures 0.117in.

upc-supplemental-2

Details on the upc-supplemental-2 bar code type.

The code represents a 2 digit decimal number that can be used in conjunction with the upc a bar code type. Typically the code is placed to the right of the upc a bar code. The gap between the rightmost bar of the upc a code and the leftmost bar of the supplemental code should not be less than 2.31mm and should not exceed 3.3mm.

The nominal size is 0.26in x 1.02in (w x h). The bar code is painted without padding at the sides.

upc-supplemental-5

Details on the bar code type.

The code represents a 5 digit decimal number that can be used in conjunction with the upc a bar code type. Typically the code is placed to the right of the ean upc a code. The gap between the rightmost bar of the upc a code and the leftmost bar of the supplemental code should not be less than 2.31mm and should not exceed 3.3mm.

The nominal size is 0.611in x 1.02in (w x h). The bar code is painted without padding at the sides.

This example shows how the supplemental code can be used in conjunction with a upc a code:

```
<LAYOUTNODE orientation="horizontal" width="min" length="min">
  <BARCODEBOX codeType="upc a" codeValue="01234567891" fontSize="10"
    mirrored="true"/>
  <BARCODEBOX codeType="upc supplemental 2" codeValue="47" fontSize="10"
    mirrored="true"/>
</LAYOUTNODE>
```


RTL Classes Overview

Object Classes

There are object classes for each type of the report item properties. The type of each property is indicated in the [Properties](#) page. The methods of these classes may be used in [RTL Expressions](#) to define a property value.

Table 195: RTL Classes

Class	Description
Boolean	Contains methods used for all logical operations.
Color	Contains methods and static member variables related to color.
Enum	A set of classes, consisting of a class for each property of this type; each class contains static member variables that provide a list of valid values for the corresponding property.
Date	Provides methods for date formatting and parsing.
FGLNumericVariable	For every 4GL numeric variable of report data, an object is created that is an instance of an FGLNumericVariable. These objects hold the value of the 4GL variable.
FGLStringVariable	For every CHAR, VARCHAR, STRING, TEXT, DATE, DATETIME and INTERVAL 4GL variable of report data, an object is created that is an instance of an FGLStringVariable. These objects hold the value of the 4GL variable.
Numeric	Contains methods used for all numeric operations. The class has the precision of a double and the arithmetic operators are defined for objects of this type.
Runtime	Provides functions that simplify the creation of dynamic designs that change behavior based on the runtime setup.
String	Contains methods used for all string operations.

Dimension Resolver Unit Names

Table 196: Unit Names

Unit abbreviations	Unit description	Point value	Example
point pt	Point value (72.27point = 1in)	1	10point
scrpixels[xy]	Screen pixel value (e.g. for 96DPI: 96pixel = 1in = 72.27point)	Depends on screen resolution of the current screen. The default value is taken from the local VM, not from a potential viewer residing on a different machine.	640scrpixelsx
prnpixels[xy]	Printer pixel value (e.g. for 300DPI 300pixel = 1in = 72.27point)	Depends on printer resolution of current printer (printer resolution defaults to 300 when there is no printer on	150prnpixelsy

Unit abbreviations	Unit description	Point value	Example
		the current pipe or when the printer is on a part of the pipe that resides on another machine).	
pica pc	Pica value (12point = 1pica)	12	3pica
inch in	Inch value (72.27point = 1in)	72.27	10in
bigpoint bp	Big point (72bigpoint = 1in)	72.27/72	10bp
cm	Centimeter value (2.54cm = 1in)	72.27/2.54	10cm
mm	Centimeter value (10mm = 1cm)	72.27/2.54/10	10mm
didot dd	Didot point (1157dd = 1238pt)	1238/1157	10dd
cicero cc	Cicero value(1cc = 12dd)	12*1238/1157	0.5cc

Paper Format Abbreviations

Table 197: Paper Format Abbreviations

Unit abbreviations	Unit Description	Width value	Length value
iso4a0(width length)	ISO 4A0	1682mm	2378mm
iso2a0(width length)	ISO 2A0	1189mm	1682mm
(isodesignatedlong dl dinlang)(width length)	ISO designated long	110mm	220mm
executive(width length)	Executive	7.25in	10.5in
(folio germanlegalfanfold)(width length)	Folio/German legal fanfold	8.5in	13in
(invoice statement)(width length)	Invoice/Statement	5.5in	8.5in
(ledger tabloid 11x7)(width length)	Ledger/Tabloid	11in	7in
(naletter letter note)(width length)	Letter	8.5in	11in
(nalegal legal)(width length)	Legal	8.5in	14in
quarto(width length)	Quarto	215mm	275mm
a(width length)	Engineering A	8.5in	11in
b(width length)	Engineering B	11in	17in

Unit abbreviations	Unit Description	Width value	Length value
c(width length)	Engineering C	17in	22in
d(width length)	Engineering D	22in	34in
e(width length)	Engineering E	34in	44in
(na10x15envelope 10x15envelope)(width length)	Envelope 10x15	10in	15in
(na10x14envelope 10x14envelope)(width length)	Envelope 10x14	10in	14in
(na10x13envelope 10x13envelope)(width length)	Envelope 10x13	10in	13in
(na9x12envelope 9x12envelope)(width length)	Envelope 9x12	9in	12in
(na9x11envelope 9x11envelope)(width length)	Envelope 9x11	9in	11in
(na7x9envelope 7x9envelope)(width length)	Envelope 7x9	7in	9in
(na6x9envelope 6x9envelope)(width length)	Envelope 6x9	6in	9in
(nanumber9envelope number9envelope env9)(width length)	Envelope number 9	3+7/8in	8+7/8in
(nanumber10envelope number10envelope env10)(width length)	Envelope number 10	1+1/8in	9+1/2in
(nanumber11envelope number11envelope env11)(width length)	Envelope number 11	4+1/2in	10+3/8in
(nanumber12envelope number12envelope env12)(width length)	Envelope number 12	4+3/4in	11in
(nanumber14envelope number14envelope env14)(width length)	Envelope number 14	5in	11+1/2in
(envinvite inviteenvelope invite)(width length)	Invite envelope	220mm	220mm
(envitaly italyenvelope italy)(width length)	Italy envelope	110mm	230mm

Unit abbreviations	Unit Description	Width value	Length value
(envmonarch monarchenvelope monarch)(width length)	Monarch envelope	3+7/8in	7+1/2in
(envpersonal personalenvelope personal)(width length)	Personal Envelope	3+5/8in	6+1/2in
(usstandardfanfold usstdfanfold)(width length)	US standard fanfold	14.875in	11in
(germanstandardfanfold germanstdfanfold)(width length)	German standard fanfold	8.5in	12in
(isoa0 a0 dina0)(width length)	ISO/DIN & JIS A0	841mm	1189mm
(isoa1 a1 dina1)(width length)	ISO/DIN & JIS A1	594mm	841mm
(isoa2 a2 dina2)(width length)	ISO/DIN & JIS A2	420mm	594mm
(isoa3 a3 dina3)(width length)	ISO/DIN & JIS A3	297mm	420mm
(isoa4 a4 dina4)(width length)	ISO/DIN & JIS A4	210mm	297mm
(isoa5 a5 dina5)(width length)	ISO/DIN & JIS A5	148mm	210mm
(isoa6 a6 dina6)(width length)	ISO/DIN & JIS A6	105mm	148mm
(isoa7 a7 dina7)(width length)	ISO/DIN & JIS A7	74mm	105mm
(isoa8 a8 dina8)(width length)	ISO/DIN & JIS A8	52mm	74mm
(isoa9 a9 dina9)(width length)	ISO/DIN & JIS A9	37mm	52mm
(isoa10 a10 dina10)(width length)	ISO/DIN & JIS A10	26mm	37mm
(isob0 b0 dinb0)(width length)	ISO/DIN B0	1000mm	1414mm
(isob1 b1 dinb1)(width length)	ISO/DIN B1	707mm	1000mm
(isob2 b2 dinb2)(width length)	ISO/DIN B2	500mm	707mm
(isob3 b3 dinb3)(width length)	ISO/DIN B3	353mm	500mm

Unit abbreviations	Unit Description	Width value	Length value
(isob4 b4 dinb4)(width length)	ISO/DIN B4	250mm	353mm
(isob5 b5 dinb5)(width length)	ISO/DIN B5	176mm	250mm
(isob6 b6 dinb6)(width length)	ISO/DIN B6	125mm	176mm
(isob7 b7 dinb7)(width length)	ISO/DIN B7	88mm	125mm
(isob8 b8 dinb8)(width length)	ISO/DIN B8	62mm	88mm
(isob9 b9 dinb9)(width length)	ISO/DIN B9	44mm	62mm
(isob10 b10 dinb10)(width length)	ISO/DIN B10	31mm	44mm
(isoc0 c0 dinc0)(width length)	ISO/DIN C0	917mm	1297mm
(isoc1 c1 dinc1)(width length)	ISO/DIN C1	648mm	917mm
(isoc2 c2 dinc2)(width length)	ISO/DIN C2	458mm	648mm
(isoc3 c3 dinc3)(width length)	ISO/DIN C3	324mm	458mm
(isoc4 c4 dinc4)(width length)	ISO/DIN C4	229mm	324mm
(isoc5 c5 dinc5)(width length)	ISO/DIN C5	162mm	229mm
(isoc6 c6 dinc6)(width length)	ISO/DIN C6	114mm	162mm
(isoc7 c7 dinc7)(width length)	ISO/DIN C7	81mm	114mm
(isoc8 c8 dinc8)(width length)	ISO/DIN C8	57mm	81mm
(isoc9 c9 dinc9)(width length)	ISO/DIN C9	40mm	57mm
(isoc10 c10 dinc10)(width length)	ISO/DIN C10	28mm	40mm
jisb0(width length)	JIS B0	1030mm	1456mm
jisb1(width length)	JIS B1	728mm	1030mm
jisb2(width length)	JIS B2	515mm	728mm
jisb3(width length)	JIS B3	364mm	515mm
jisb4(width length)	JIS B4	257mm	364mm

Unit abbreviations	Unit Description	Width value	Length value
jjsb5(width length)	JIS B5	182mm	257mm
jjsb6(width length)	JIS B6	128mm	182mm
jjsb7(width length)	JIS B7	91mm	128mm
jjsb8(width length)	JIS B8	64mm	91mm
jjsb9(width length)	JIS B9	45mm	64mm
jjsb10(width length)	JIS B10	32mm	45mm

Customize Report Designer: preferences

Document View

- The **Prefer to display item name over RTL expression text** checkbox: When selected, user-defined labels are displayed instead of the expressions. A label is considered user-defined if it does not match the generated name "[NodeType][Index]" (e.g. "WordBox12").

Paper Settings Preferences

You can specify the default paper settings to be used for report design documents.

- **Orientation:** portrait, landscape
- **Units:** centimeter, inch
- **Page Size Format:** standard - choose a value from the combobox, custom
- **Margins** - left, right, top, bottom

RTL Class Reference

- [The Boolean Class](#) on page 806
- [The Color Class](#) on page 807
- [The Date Class](#) on page 810
- [The Enum Classes](#) on page 812
- [The FGLNumericVariable Class](#) on page 819
- [The FGLStringVariable Class](#) on page 820
- [The Numeric Class](#) on page 821
- [The Runtime Class](#) on page 831
- [The String Class](#) on page 834

The Boolean Class

Details about the Boolean class and its public members.

- [Syntax](#)
- [Member Objects](#)
- [Usage](#)

Syntax

```
Boolean
```

Member Objects

Table 198: Member Objects (Static Member Variables)

Name	Description
TRUE	Contains the value TRUE.
FALSE	Contains the value FALSE.

Usage

With RTL classes, it is not possible to create and subclass objects. The `new` keyword is not supported.

Member Objects Usage

All expressions in the RTL expression language that contain [relational operators](#) return one of these objects, which are static member variables that do not require an object instance.

To specify `TRUE` or `FALSE` in a formula, the variables are prefixed with the 'Boolean' class name and the `'` character.

Example

```
Boolean.TRUE
```

The Color Class

The Color class provides methods for specifying the color of an object.

- [Syntax](#)
- [Member Objects](#)
- [Class Methods](#)
- [Object Methods](#)

Syntax

```
Color
```

Member Objects: Static Member Variables

These member variables do not require an object. They are prefixed by the Color class name.

Table 199: Member Objects (Static Member Variables)

Name	Description
BLACK	Specifies a Color object having the color BLACK
BLUE	Specifies a Color object having the color BLUE
CYAN	Specifies a Color object having the color CYAN
DARK_GRAY	Specifies a Color object having the color DARK_GRAY
GRAY	Specifies a Color object having the color GRAY
GREEN	Specifies a Color object having the color GREEN

Name	Description
LIGHT_GRAY	Specifies a Color object having the color LIGHT_GRAY
MAGENTA	Specifies a Color object having the color MAGENTA
ORANGE	Specifies a Color object having the color ORANGE
PINK	Specifies a Color object having the color PINK
RED	Specifies a Color object having the color RED
WHITE	Specifies a Color object having the color WHITE
YELLOW	Specifies a Color object having the color YELLOW

Usage

With RTL classes, it is not possible to create and subclass objects. The `new` keyword is not supported.

The default color for a report item, such as a `LayoutNode`, is black. You can change the color of the item by entering a value for the `color` or `bgColor` property in the [Properties View](#). These properties are of type `Color`; instead of selecting a color from the Edit Expressions color palette, you can use the members of this class in an expression that returns a `Color` object.

Member Objects Usage

These objects are static member variables that do not require an object instance. The variables are prefixed with the `Color` class name and the `'.'` character.

Example:

```
Color.RED
```

Class Methods

Class methods do not require a `Color` object instance. When you invoke a class method, it is prefixed with the `Color` class name and the `'.'` character.

Table 200: Class Methods (Static Member Methods)

Name	Description
<code>fromHSBA(h Numeric, s Numeric, b Numeric)</code>	<p>Returns a <code>Color</code> object based on the specified values of hue, saturation, and brightness for the HSB color model.</p> <pre>Color.fromHSBA(h,s,b)</pre> <ul style="list-style-type: none"> <code>h</code> (hue) is any floating-point number. The floor of this number is subtracted from it to create a fraction between 0 and 1, which is then multiplied by 360. <code>s</code> (saturation) is a floating-point value between zero and one (numbers within the range 0.0-1.0).

Name	Description
	<ul style="list-style-type: none"> <i>b</i> (brightness) is a floating-point value between zero and one (numbers within the range 0.0-1.0).
<pre>fromHSBA(h Numeric, s Numeric, b Numeric, a Numeric)</pre>	<p>Returns a Color object based on the specified values of hue, saturation, brightness, and alpha for the HSB color model.</p> <pre>Color.fromHSBA(h,s,b,a)</pre> <ul style="list-style-type: none"> <i>h</i> (hue) is any floating-point number. The floor of this number is subtracted from it to create a fraction between 0 and 1, which is then multiplied by 360. <i>s</i> (saturation) is a floating-point value between zero and one (numbers within the range 0.0-1.0). <i>b</i> (brightness) is a floating-point value between zero and one (numbers within the range 0.0-1.0). <i>a</i> (alpha) is the alpha component.
<pre>fromRGBA(r Numeric, g Numeric, b Numeric)</pre>	<p>Returns an opaque sRGB Color object with the specified red, green, and blue values.</p> <pre>Color.fromRGBA(r,g,b)</pre> <ul style="list-style-type: none"> <i>r</i> (red) is within the range (0.0 - 255) <i>g</i> (green) is within the range (0.0 - 255) <i>b</i> (blue) is within the range (0.0 - 255) <p>Alpha defaults to 1.0. The color used depends on the best match from the colors available for the output device.</p>
<pre>fromRGBA(r Numeric, g Numeric, b Numeric, a Numeric)</pre>	<p>Returns an sRGB Color object with the specified red, green, blue, and alpha values.</p> <pre>fromRGBA(r,g,b,a)</pre> <ul style="list-style-type: none"> <i>r</i> (red) is within the range (0.0 - 255) <i>g</i> (green) is within the range (0.0 - 255) <i>b</i> (blue) is within the range (0.0 - 255) <i>a</i> (alpha) is within the range (0.0 - 255) <p>The color used depends on the best match from the colors available for the output device.</p>

Example - Color Class Method

```
Color.fromRGBA(0.5,0.5,0.5,0.5)
```

Object Methods

Color object methods are called on a Color object instance. Object methods require an object reference. When you invoke the method, it is prefixed with the object instance name and the "." character.

Table 201: Object Methods

Name	Description
<code>brighter()</code>	Brightens the Color object. Creates a brighter version of the Color object by applying an arbitrary scale factor to each of the RGB components. Invoking a series of invocations of the darker and brighter methods might give an inconsistent result because of rounding errors.
<code>darker()</code>	Darkens the Color object. Creates a darker version of the Color object by applying an arbitrary scale factor to each of the RGB components. Invoking a series of invocations of the darker and brighter methods might give an inconsistent result because of rounding errors.
<code>getAlpha()</code>	Returns the Numeric alpha component in the range 0-255.
<code>getBrightness()</code>	Returns the Numeric brightness value of the value in the HSB color model
<code>getBlue()</code>	Returns the Numeric blue component in the range 0-255 in the default sRGB space.
<code>getGreen()</code>	Returns the Numeric green component in the range 0-255 in the default sRGB space.
<code>getRGBA()</code>	Returns the Numeric RGB value representing the color in the default sRGB color model. Bits 24-31 are alpha, 16-23 are red, 8-15 are green, and 0-7 are blue.
<code>getHue()</code>	Returns the hue value of the value in the HSB color model.
<code>getRed()</code>	Returns the Numeric red component in the range 0-255 in the default sRGB space.
<code>getSaturation()</code>	Returns the saturation value of the value in the HSB color model.
<code>toString()</code>	Return a string representation in the form "#argb".

Example - Color Object Method

```
Color.RED.darker()
```

The Date Class

The Date class provides methods for date formatting and parsing.

- [Syntax](#)

- [Methods](#)
- [Usage](#)
- [Formatting Symbols for Dates](#)

Syntax

```
Date
```

Methods

Class methods are static methods that do not require an object. The method name is prefixed by the class name.

Table 202: Class Methods (Static Member Methods)

Name	Description
<code>fromIsoValue(String value)</code>	Constructs a Date from the date value; returns a Date instance representing the value. The value is expected in iso format ("YYYY-MM-DD"). Using the function guarantees that the Date value is constructed correctly without the danger of runtime parse errors due to changing 4GL date formatting or locale settings.
<code>parseString(String value, String format)</code>	Constructs a Date from <i>value</i> . The parsing is based on the passed <i>format</i> pattern. See Formatting Symbols for Dates .
<code>format(java.lang.String.format)</code>	Formats a Date as a String according to a format specification. The format specification is 4GL-compatible. Returns a string representation of the date. See Formatting Symbols for Dates .
<code>today()</code>	Constructs a Date from the current date value. Returns a Date instance representing the current date.

Usage

With RTL classes, it is not possible to create and subclass objects. The `new` keyword is not supported.

These static methods do not require a Date object instance. When you invoke the method, it is prefixed with the `Date` class name and the `'.'` For example, this expression uses the `isoValue` of the variable `orderline.orders.orderdate` to create a valid date for this value in the specified format:

```
Date.fromIsoValue(orderline.orders.orderdate.isoValue).format("DDD DD MMM YYYY")
```

Formatting symbols

The formatting symbol can use either uppercase or lowercase letters.

Table 203: Formatting Symbols for Dates

Character	Description
dd	Day of the month as a two-digit integer.

Character	Description
ddd	Three-letter English-language abbreviation of the day of the week, for example, Mon, Tue.
mm	Month as a two-digit integer.
mmm	Three-letter English-language abbreviation of the month, for example, Jan, Feb.
YY	Year, as a two-digit integer representing the two trailing digits.
YYYY	Year as a four-digit number.

The Enum Classes

These classes provide the list of valid values for form item properties that are of type Enum.

- [The Alignment Class](#) on page 812
- [The TextAlignment Class](#) on page 813
- [The BaselineType Class](#) on page 813
- [The LayoutDirection Class](#) on page 813
- [The Y-SizeAdjustment Class](#) on page 814
- [The PageNoFormat Class](#) on page 814
- [The TrimText Class](#) on page 815
- [The X-SizeAdjustment Class](#) on page 815
- [FloatingBehavior Class](#) on page 815
- [Section Class](#) on page 816
- [XYChartDrawAs Class](#) on page 816
- [MapChartDrawAs Class](#) on page 817
- [CategoryChartDrawAs Class](#) on page 817
- [CodeType Class](#) on page 817
- [BorderStyles Classes](#) on page 819

The Alignment Class

Public static member variables that represent the valid values for the Alignment property.

Syntax

```
Alignment
```

This class consists of a set of public static member variables that represent the valid values for the [alignment](#) property.

Table 204: Member Objects (Static Member Variables)

Name
Baseline
Center
Far
Near
None

The TextAlignment Class

Public static member variables that represent the valid values for the textAlignment property.

Syntax

```
TextAlignment
```

The class consists of a set of public static member variables that represent the valid values for the [textAlignment](#) property.

Table 205: Member Objects (Static Member Variables)

Name
Center
Left
Right

The BaselineType Class

public static member variables that represent the valid values for the baselineType property.

Syntax

```
BaselineType
```

The class consists of a set of public static member variables that represent the valid values for the [baselineType](#) property.

Table 206: Member Objects (Static Member Variables)

Name
Leftleft
Leftright
Rightleft
Rightright

The LayoutDirection Class

Public static member variables that represent the valid values for the layoutDirection property.

Syntax

```
LayoutDirection
```

The class consists of a set of public static member variables that represent the valid values for the [layoutDirection](#) property.

Table 207: Member Objects (Static Member Variables)

Name
BottomToTop
HorizontalNatural

Name
LeftToRight
RightToLeft
Swapped
TopToBottom
TurnLeft
TurnRight
Unturned
UpsideDown
VerticalNatural
VerticalUnnatural

The Y-SizeAdjustment Class

Public static member variables that represent the valid values for the Y-SizeAdjustment property.

Syntax

```
Y-SizeAdjustment
```

The class consists of a set of public static member variables that represent the valid values for the [Y-SizeAdjustment](#) property.

Table 208: Member Objects (Static Member Variables)

Name
ExpandtoParent
ShrinktoChildren

The PageNoFormat Class

Public static member variables that represent the valid values for the pageNoFormat property.

Syntax

```
PageNoFormat
```

The class consists of a set of public static member variables that represent the valid values for the [pageNoFormat](#) property.

Table 209: Member Objects (Static Member Variables)

Name
Arabic
Lowerroman
Upperroman

The TrimText Class

Public static member variables that represent the valid values for the trimText property.

Syntax

```
TrimText
```

The class consists of a set of public static member variables that represent the valid values for the [trimText](#) property.

Table 210: Member Objects (Static Member Variables)

Name
Both
Compress
Left
Right

The X-SizeAdjustment Class

Public static member variables that represent the valid values for the X-SizeAdjustment property.

Syntax

```
X-SizeAdjustment
```

The class consists of a set of public static member variables that represent the valid values for the [X-SizeAdjustment](#) property.

Table 211: Member Objects (Static Member Variables)

Name
ExpandtoParent
ShrinktoChildren

FloatingBehavior Class

Public static member variables that represent the valid values for the floatingBehavior property.

Syntax

```
FloatingBehavior
```

The class consists of a set of public static member variables that represent the valid values for the property.

Table 212: Member Objects (Static Member Variables)

Name
Encloses
Free

Section Class

Public static member variables that represent the valid values for the Section property.

Syntax

```
Section
```

The class consists of a set of public static member variables that represent the valid values for the property.

Table 213: Member Objects (Static Member Variables)

Name
AnyPageFooter
AnyPageHeader
EvenPageHeader
EvenPageFooter
FirstPageHeader
FirstPageFooter
OddPageFooter
OddPageHeader
ItemSeparator

XYChartDrawAs Class

Public static member variables that represent the valid values for the XY chart DrawAs property.

Syntax

```
XYChartDrawAs
```

The class consists of a set of public static member variables that represent the valid values for the property.

Table 214: Member Objects (Static Member Variables)

Name
Polar
Scatter
Area
StackedArea
Line
Step
StepArea
TimeSeries

MapChartDrawAs Class

Public static member variables that represent the valid values for the map chart DrawAs property.

Syntax

```
MapChartDrawAs
```

The class consists of a set of public static member variables that represent the valid values for the property.

Table 215: Member Objects (Static Member Variables)

Name
Bar
Bar3D
Pie
Pie3D
Ring

CategoryChartDrawAs Class

Public static member variables that represent the valid values for the category chart DrawAs property.

Syntax

```
CategoryChartDrawAs
```

The class consists of a set of public static member variables that represent the valid values for the property.

Table 216: Member Objects (Static Member Variables)

Name
Area
Bar
Bar3D
Line
Line3D
StackedArea
StackedBar
Waterfall

CodeType Class

Public static member variables that represent the valid values for the codeType property.

Syntax

```
CodeType
```

The class consists of a set of public static member variables that represent the valid values for the property.

Table 217: Member Objects (Static Member Variables)

Name
Upc_a
Upc_e
Upc_supplemental_2
Upc_supplemental_5
Gs1_13
Ean_13
Gs1-8
Ean_8
Gs1_supplemental_2
Ean_supplemental_2
Gs1_supplemental_5
Ean_supplemental_5
Code_128
Gs1_code_128
Ean_Code_128
Code_2_5_industrial
Code_2_5_inverted
Code_2_5_IATA
Code_2_5_interleaved
Code_2_5_matrix
Code_2_5_datalogic
Code_BCD_matrix
Code_11_matrix
Code_39
Code_39_extended
Code_32
Code_93
Code_93_extended
Codabar_18
Codabar_2
Pdf_417

Name
Data_matrix
Gs1_data_matrix
Ean_data_matrix
Qr_code
Intelligent_mail

BorderStyle Classes

Public member variables that represent the valid values for the various border style properties.

Syntax

The attributes **borderStyle**, **borderTopStyle**, **borderRightStyle**, **borderBottomStyle** and **borderLeftStyle** have a set of public member variables that represent the valid values for the property:

Table 218: Member Objects (Static Member Variables)

Name
Dashed
Dotted
Double
Groove
Inset
Outset
Ridge
solid

The FGLNumericVariable Class

For every Genero numeric variable (INTEGER, SMALLINT, FLOAT, SMALLFLOAT, DECIMAL and MONEY) of report data, an object is created that is an instance of an FGLNumericVariable. These objects hold the value of the 4GL variable.

- [Syntax](#) on page 819
- [Member Objects](#) on page 819
- [Class Methods](#) on page 820
- [Class Usage](#) on page 820

Syntax

```
FGLStringVariable
```

Member Objects

Defined fields for the FGLNumericVariable class.

Table 219: Member Objects (Member Variables)

Name	Description
value	A Numeric containing the value of the field; this is the same as the value of the 4GL variable. See Usage .
fglValue	A String containing the value of the field as formatted by the DVM. See Usage .
name	A String specifying the name of the field.
caption	A String specifying the title of the field.
type	A String specifying the type of the field.

Class Methods

This subclass inherits the public methods of the [Numeric class](#).

Class Usage

In addition to the value of the 4GL variable, these objects contain member variables, among which is the variable `value` also containing the numeric value. For this reason, it is legal to write `"order_line.itemprice"` in your expression as a shortcut for `"order_line.itemprice.value"`.

The member variable `fglValue` contains a String representing the numeric value as formatted by the DVM, taking into consideration such parameters as USING, DBMONEY, etc. In contrast, the member variable `value` is a numeric value without formatting.

The FGLStringVariable Class

For every CHAR, VARCHAR, STRING, TEXT, DATE, DATETIME and INTERVAL Genero variable of report data, an object is created that is an instance of an `FGLStringVariable`. These objects hold the value of the Genero variable.

- [Syntax](#)
- [Member Objects](#) on page 820
- [Class Methods](#) on page 821
- [Class Usage](#) on page 821

Syntax

```
FGLStringVariable
```

Member Objects

Defined fields for the `FGLStringVariable` class.

Table 220: Member Objects (Member Variables)

Name	Description
value	A String containing the value of the field; this is the same as the value of the Genero variable. See Usage .
name	A String specifying the name of the field.
caption	A String specifying the title of the field.
type	A String specifying the type of the field.

Class Methods

This subclass inherits the public methods of the [String class](#).

Class Usage

In addition to the value of the Genero variable, these objects contain a member variable "[value](#)" which also contains the value. For this reason, it is legal to write "order_line.billState" in your expression as a shortcut for "order_line.billState.value".

The Numeric Class

Details about the Numeric class and its public members.

Values for this data type are limited to 15 significant digits.

- [Syntax](#)
- [Methods](#)
- [Usage](#)
- [Examples](#)

Syntax

```
Numeric
```

Methods

Table 221: Object Methods

Name	Description
<code>abs()</code>	Returns the absolute value.
<code>atan2(Numeric x)</code>	Returns the angle <i>theta</i> from the conversion of rectangular coordinates (x, y) to polar coordinates (r, <i>theta</i>)
<code>byteValue()</code>	Returns the value converted to a byte.
<code>cbrt()</code>	Returns the cube root of a numeric value.
<code>ceil()</code>	Returns the smallest numeric value that is greater than or equal to the value and is equal to a mathematical integer.
<code>cos()</code>	Returns the trigonometric cosine of an angle.
<code>cosh()</code>	Returns the hyperbolic cosine of a numeric value.
<code>exp()</code>	Returns the base -e exponential.
<code>floor()</code>	Returns the largest numeric value that is less than or equal to the value and is equal to a mathematical integer.
<code>format("format-string")</code>	<p>Converts the value to a string representation defined by a format string.</p> <p>For example, for DECIMAL and FLOAT data types, <i>format-string</i> consists of pound signs (#) that represent digits and a decimal point; that is, "###.###" produces three places to the left of the decimal point</p>

Name	Description
	and exactly two to the right. Additional options for the string are listed in the Usage section, format.
<code>getExponent()</code>	Returns the unbiased exponent used in the representation of a numeric value.
<code>intValue()</code>	Returns the value converted to an integer.
<code>isInfinite()</code>	Returns TRUE if the value has infinite value
<code>isNaN()</code>	Returns TRUE if the value is not a number
<code>isNull()</code>	Returns true if <code>toDouble()</code> is 0 and the object is tagged as null, otherwise false. This is the case for null valued input variables read from the input stream. For backward compatibility, null values do not have special behavior when used with the various

Name	Description
	operators. Specifically a numeric input variable that is null behaves in arithmetic like the 0 value.
<code>log()</code>	Returns the natural logarithm (base e) of a numeric value.
<code>log10()</code>	Returns the base 10 logarithm of a numeric value.
<code>max(Numeric b)</code>	Returns the greater of two values.
<code>min(Numeric b)</code>	Returns the smaller of two values.
<code>rint()</code>	Returns the Numeric value that is closest in value to the value and is equal to a mathematical integer.
<code>round()</code>	Returns the closes integer to the value.
<code>signum()</code>	Returns the signum function of the value; zero if the value is zero, 1.0 if the value is greater than zero, -1.0 if the value is less than zero.
<code>sin()</code>	Returns the trigonometric sine, measured in radians.
<code>sinh()</code>	Returns the hyperbolic sine of a numeric value.
<code>sqrt()</code>	Returns the tangent of an angle measured in radians.
<code>tan()</code>	Returns the tangent of an angle measured in radians.
<code>tanh()</code>	Returns the hyperbolic tangent of a numeric value.
<code>toBoolean()</code>	Returns the Boolean false when the value is 0. Returns true for any other value.
<code>toChar()</code>	Returns the unicode character representation of a numeric value.
<code>toColor()</code>	Returns a color object. The value is interpreted as a RGB integer.
<code>toDegrees()</code>	Converts the value from radians to degrees.
<code>toRadians()</code>	Converts the value from degrees to radians.
<code>toString()</code>	Converts the value to a string representation.
<code>getPhysicalPageNumber()</code>	Gets the current page number of the physical page.

Usage

Important: This data type is limited to 15 significant digits.

With RTL classes, it is not possible to create and subclass objects. The `new` keyword is not supported.

abs()

```
abs ( )
```

Returns the absolute value of an int value:

- If the value is not negative, the value is returned.
- If the value is negative, the negation of the value is returned.
- If the value is equal to the value of Integer. MIN_VALUE, the most negative representable int value, the result is that same value, which is negative.

cos()

```
cos ( )
```

Returns the trigonometric cosine of an angle. However, If the value is NaN or an infinity, then the result is NaN.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

sin()

```
sin ( )
```

Returns the trigonometric sine of an angle. However,

- If the value is NaN or an infinity, then the result is NaN.
- If the value is zero, then the result is a zero with the same sign as the value.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

tan()

```
tan ( )
```

Returns the trigonometric tangent of an angle. Special cases:

- If the value is NaN or an infinity, then the result is NaN.
- If the value is zero, then the result is a zero with the same sign as the value.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

byteValue()

```
byteValue ( )
```

Returns the value converted to a byte .

cbrt()

```
cbrt ( )
```

Returns the cube root of a double value. For positive finite x , $\text{cbrt}(-x) == -\text{cbrt}(x)$; that is, the cube root of a negative value is the negative of the cube root of that value's magnitude.

Special cases:

- If the value is NaN, then the result is NaN.
- If the value is infinite, then the result is an infinity with the same sign as the value.
- If the value is zero, then the result is a zero with the same sign as the value.

The computed result must be within 1 ulp of the exact result.

ceil()

```
ceil()
```

Returns the smallest (closest to negative infinity) double value that is greater than or equal to the value and is equal to a mathematical integer.

Special cases:

- If the value is already equal to a mathematical integer, then the result is the same as the value.
- If the value is NaN or an infinity or positive zero or negative zero, then the result is the same as the value.
- If the value is less than zero but greater than -1.0, then the result is negative zero.

Note that the value of $\text{Math.ceil}(x)$ is exactly the value of $-\text{Math.floor}(-x)$.

cosh()

```
cosh()
```

Returns the hyperbolic cosine of a double value. The hyperbolic cosine of x is defined to be $(e^x + e^{-x})/2$ where e is Euler's number.

Special cases:

- If the value is NaN, then the result is NaN.
- If the value is infinite, then the result is positive infinity.
- If the value is zero, then the result is 1.0.

The computed result must be within 2.5 ulps of the exact result.

exp()

```
exp()
```

Returns Euler's number e raised to the power of a double value. Special cases:

- If the value is NaN, the result is NaN.
- If the value is positive infinity, then the result is positive infinity.
- If the value is negative infinity, then the result is positive zero.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

floor()

```
floor()
```

Returns the largest (closest to positive infinity) double value that is less than or equal to the value and is equal to a mathematical integer.

Special cases:

- If the value is already equal to a mathematical integer, then the result is the same as the value.
- If the value is NaN or an infinity or positive zero or negative zero, then the result is the same as the value.

getExponent()

```
getExponent()
```

Returns the unbiased exponent used in the representation of a float.

Special cases:

- If the value is NaN or infinite, then the result is Float. MAX_EXPONENT + 1.
- If the value is zero or subnormal, then the result is Float. MIN_EXPONENT -1.

intValue()

```
intValue()
```

returns the value converted to an integer (signed 32 bit)

isInfinite()

```
isInfinite()
```

returns the Boolean value true in case the value has the infinite value

isNaN()

```
isNaN()
```

returns the Boolean value true in case that the value is not a number

log()

```
log()
```

Returns the natural logarithm (base e) of a double value.

Special cases:

- If the value is NaN or less than zero, then the result is NaN.
- If the value is positive infinity, then the result is positive infinity.
- If the value is positive zero or negative zero, then the result is negative infinity.

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

log10()

```
log10()
```

Returns the base 10 logarithm of a double value.

Special cases:

- If the value is NaN or less than zero, then the result is NaN.
- If the value is positive infinity, then the result is positive infinity.

- If the value is positive zero or negative zero, then the result is negative infinity.
- If the value is equal to $10n$ for integer n , then the result is n .

The computed result must be within 1 ulp of the exact result. Results must be semi-monotonic.

rint()

```
rint()
```

Returns the double (floating-point) value that is closest in value to the value and is equal to a mathematical integer. If two double values that are mathematical integers are equally close, the result is the integer value that is even.

Special cases:

- If the value is already equal to a mathematical integer, then the result is the same as the value.
- If the value is NaN or an infinity or positive zero or negative zero, then the result is the same as the value.

Returns:

the closest floating-point value to a that is equal to a mathematical integer.

round()

```
round()
```

Returns the closest int to the value. The result is rounded to an integer by adding $1/2$, taking the floor of the result, and casting the result to type int. In other words, the result is equal to the value of the expression:

```
(int)Math.floor(a + 0.5f)
```

Special cases:

- If the value is NaN, the result is 0.
- If the value is negative infinity or any value less than or equal to the value of Integer.MIN_VALUE, the result is equal to the value of Integer.MIN_VALUE.
- If the value is positive infinity or any value greater than or equal to the value of Integer.MAX_VALUE, the result is equal to the value of Integer.MAX_VALUE.

See Also:

Integer.MAX_VALUE, Integer.MIN_VALUE

signum ()

```
signum ()
```

Returns the signum function of the value; zero if the value is zero, 1.0 if the value is greater than zero, -1.0 if the value is less than zero.

Special Cases:

- If the value is NaN, then the result is NaN.
- If the value is positive zero or negative zero, then the result is the same as the value.

Returns:

the signum function of the value

sinh()

```
sinh()
```

Returns the hyperbolic sine of a double value. The hyperbolic sine of x is defined to be $(e^x - e^{-x})/2$ where e is Euler's number.

Special cases:

- If the value is NaN, then the result is NaN.
- If the value is infinite, then the result is an infinity with the same sign as the value.
- If the value is zero, then the result is a zero with the same sign as the value.

The computed result must be within 2.5 ulps of the exact result.

sqrt()

```
sqrt()
```

Returns the correctly rounded positive square root of a double value.

Special cases:

- If the value is NaN or less than zero, then the result is NaN.
- If the value is positive infinity, then the result is positive infinity.
- If the value is positive zero or negative zero, then the result is the same as the value.

Otherwise, the result is the double value closest to the true mathematical square root of the value

tanh()

```
tanh()
```

Returns the hyperbolic tangent of a double value. The hyperbolic tangent of x is defined to be $(e^x - e^{-x})/(e^x + e^{-x})$, in other words, $\sinh(x)/\cosh(x)$. Note that the absolute value of the exact tanh is always less than 1.

Special cases:

- If the value is NaN, then the result is NaN.
- If the value is zero, then the result is a zero with the same sign as the value.
- If the value is positive infinity, then the result is +1.0.
- If the value is negative infinity, then the result is -1.0.

The computed result must be within 2.5 ulps of the exact result. The result of tanh for any finite input must have an absolute value less than or equal to 1. Note that once the exact result of tanh is within 1/2 of an ulp of the limit value of ± 1 , correctly signed ± 1.0 should be returned.

toChar()

```
toChar()
```

Converts the value to a unicode character representation.

For example, `65.toChar()` yields "A".

atan2(Numeric x)

```
atan2(Numeric x)
```

Returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta). This method computes the phase theta by computing an arc tangent of y/x in the range of -pi to pi.

Special cases:

- If either value is NaN, then the result is NaN.
- If the value is positive zero and the argument is positive, or the value is positive and finite and the argument is positive infinity, then the result is positive zero.
- If the value is negative zero and the argument is positive, or the value is negative and finite and the argument is positive infinity, then the result is negative zero.
- If the value is positive zero and the argument is negative, or the value is positive and finite and the argument is negative infinity, then the result is the double value closest to pi.
- If the value is negative zero and the argument is negative, or the value is negative and finite and the argument is negative infinity, then the result is the double value closest to -pi.
- If the value is positive and the argument is positive zero or negative zero, or the value is positive infinity and the argument is finite, then the result is the double value closest to pi/2.
- If the value is negative and the argument is positive zero or negative zero, or the value is negative infinity and the argument is finite, then the result is the double value closest to -pi/2.
- If both value and argument are positive infinity, then the result is the double value closest to pi/4.
- If the value is positive infinity and the argument is negative infinity, then the result is the double value closest to 3*pi/4.
- If the value is negative infinity and the argument is positive infinity, then the result is the double value closest to -pi/4.
- If both value and argument are negative infinity, then the result is the double value closest to -3*pi/4.

The computed result must be within 2 ulps of the exact result. Results must be semi-monotonic.

Parameters:

x - the abscissa coordinate

max(Numeric b)

```
max( b )
```

Returns the greater of two int values. That is, the result is the argument closer to the value of Integer.MAX_VALUE. If the argument has the same value as the object's value the result is that same value.

Parameters:

- *b* - an argument.

See Also: Long.MAX_VALUE

min(Numeric b)

```
min( b )
```

Returns the smaller of two int values. That is, the result is the argument closer to the value of Integer.MIN_VALUE. If the argument has the same value as the objects value the result is that same value.

Parameters:

- *b* - an argument.

See Also: Long.MIN_VALUE

format("format-string")

```
format("format-string")
```

Converts the value to a string representation defined by a format string. The format string syntax is compatible to the D4GL "USING" format string. The formatting takes the values of the environment variables DBFORMAT and DBMONEY into account.

Table 222: Formatting symbols

Character	Description
*	Fills with asterisks any position that would otherwise be blank.
&	Fills with zeros any position that would otherwise be blank.
#	This does not change any blank positions in the display.
<	Causes left alignment.
, (comma)	Defines the position of the thousands separator. The thousands separator is not displayed if there are no digits to the left. By default, the thousands separator is a comma, but it can be another character as defined by DBFORMAT.
. (period)	Defines the position of the decimal separator. Only a single decimal separator may be specified. By default, the decimal separator is a period, however it can be another character as defined by DBMONEY or DBFORMAT.
-	Displays a minus sign for negative numbers.
\$	This is the placeholder for the front specification of DBMONEY or DBFORMAT.
(Displayed as left parentheses for negative numbers (accounting parentheses).
)	Displayed as right parentheses for negative numbers (accounting parentheses).

Examples

1. This example converts the number 65 to "A", its unicode character representation: `65.toChar()`
2. This example formats the numeric value of overall total as a string:
`overalltotal.format("###.##")`

The Runtime Class

The Runtime class provides methods that simplify the creation of dynamic designs that change behavior based on the runtime setup.

- [Syntax](#)
- [Methods](#)
- [Usage](#)

Syntax

Runtime

Methods

Class methods are static methods that do not require an object. The method name is prefixed by the class name.

Table 223: Class Methods (Static Member Methods)

Name	Description
<code>getDebugLevel()</code>	Numeric, returns the current debug level specified in the environment variable GREDEBUG, or 0 if no debug level was set.
<code>getEnvironmentVariable(String <i>variableName</i>)</code>	Returns a String containing the value of the specified environment variable.
<code>getOutputDeviceName()</code>	Returns a STRING indicating the output device name selected in the API call <code>fgl_report_selectDevice</code> . Can be one of the following: <ul style="list-style-type: none"> • PDF • XLS • XLSX • HTML • Image • Printer • Postscript • SVG • Browser • RTF • OORTF
<code>getPrinterMediaName()</code>	Returns a STRING indicating the media name specified in the API call <code>fgl_report_setPrinterMediaName</code> .
<code>getPrinterMediaSizeName()</code>	Returns a STRING indicating the media size name specified in the API call <code>fgl_report_setPrinterMediaSizeName</code> .
<code>getPrinterMediaTray()</code>	Returns a STRING indicating the media tray name specified in the API call <code>fgl_report_setPrinterMediaTray</code> .
<code>getPrinterName()</code>	Returns a STRING indicating the printer name specified in the API call <code>fgl_report_setPrinterName</code> .
<code>getSVGPaperSource()</code>	Returns a STRING indicating the paper source specified in the API call <code>fgl_report_setSVGPaperSource</code> ; can be: <ul style="list-style-type: none"> "Auto", "Cassette", "Envelope", "EnvelopeManual", "FormSource", "LargeCapacity", "LargeFormat",

Name	Description
	"Lower", "Middle", "Manual", "OnlyOne", "Tractor" or "SmallFormat"
getSVGPrinterName()	Returns a STRING indicating the printer name specified in the API call fgl_report_setSVGPrinterName.
producingBrowserOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "Browser"; otherwise FALSE.
producingExcelOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "XLS" or "XLSX"; otherwise FALSE.
producingHTMLOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "HTML"; otherwise FALSE.
producingImageOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "Image"; otherwise FALSE.
producingOORTFOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "OORTF"; otherwise FALSE.
producingPDFOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "PDF"; otherwise FALSE.
producingPostscriptOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "Postscript"; otherwise FALSE.
producingRTFOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "RTF"; otherwise FALSE.
producingSVGOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "SVG"; otherwise FALSE.
producingXLSOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "XLS"; otherwise FALSE.
producingXLSXOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "XLSX"; otherwise FALSE.
producingPrinterOutput()	BOOLEAN; returns TRUE if the device name selected in the API call fgl_report_selectDevice was "Printer"; otherwise FALSE.
producingForPreview()	BOOLEAN; returns TRUE if preview was selected in the API call fgl_report_selectPreview; otherwise FALSE.
xlsPagesAreMerged()	BOOLEAN, returns TRUE if page merging was selected in the API call fgl_report_configureXLSDevice; otherwise FALSE.

Usage

The class provides a number of methods that simplify the job of creating dynamic designs that change behavior based on the runtime setup.

With RTL classes, it is not possible to create and subclass objects. The `new` keyword is not supported.

The methods can be used from within RTL expressions. Some common uses might be:

- To suppress headers and footers when Excel output is selected.
- To conditionally Insert a logo based on the printer tray is selected.
- To set a background color when debugging is enabled

These static methods do not require a Runtime object instance. When you invoke the method, it is prefixed with the Runtime class name and the '!'. For example, you can suppress a header by setting its "[visibilityCondition](#)" property to this expression:

```
"!Runtime.producingExcelOutput()"
```

The String Class

Details about the String class and its public members.

- [Syntax](#)
- [Methods](#)
- [Usage and Examples](#)

Syntax

```
String
```

Methods

Table 224: Object Methods

Name	Description
<code>charAt(index Numeric)</code> RETURNING <i>result</i> String	Returns a String containing the character at the specified <i>index</i> in the current String.
<code>contains(s String)</code> RETURNING <i>result</i> Boolean	Returns a Boolean value (TRUE/FALSE) specifying whether <i>s</i> is contained within the current String.
<code>endsWith(s String)</code> RETURNING <i>result</i> Boolean	Returns a Boolean value (TRUE/FALSE) specifying whether the current String ends with <i>s</i> .
<code>equals(s String)</code> RETURNING <i>result</i> Boolean	Returns a Boolean value (TRUE/FALSE) specifying whether <i>s</i> matches the current String. If one of the Strings is null the method returns FALSE.
<code>equalsIgnoreCase(s String)</code> RETURNING <i>result</i> Boolean	Returns a Boolean value (TRUE/FALSE) specifying whether <i>s</i> matches the current String, ignoring character case. If one of the Strings is null the method returns FALSE.
<code>format (number Numeric, format Enum)</code>	Sets the format of the page number string for a PAGENOBOX . The value for format can be: ARABIC, LOWERROMAN, UPPERROMAN

Name	Description
RETURNING <i>result</i> String	
<code>indexOf(s String)</code> RETURNING <i>result</i> Numeric	Returns a Numeric value representing the index of <i>s</i> within the current String.
<code>indexOf(s String , index Numeric)</code> RETURNING <i>result</i> Numeric	Returns a Numeric value representing the index of <i>s</i> within the current String, starting from byte position <i>index</i> . Returns zero if the substring was not found. Returns -1 if the current String is null.
<code>isEmpty()</code> RETURNING <i>result</i> Boolean	Returns a Boolean value. Returns true if the current string has a length of zero (<code>length()=0</code>), otherwise false.
<code>isNull()</code> RETURNING <i>result</i> Boolean	<p>Returns a Boolean value. Returns true if the current string has a length of zero (<code>length()=0</code>) and the string is tagged as null, otherwise false. This is the case for null valued input variables read from the input stream.</p> <p>For backward compatibility, null values do not have special behavior when used with the various operators. Specifically an input variable of type string that is null behaves like the empty string.</p>
<code>lastIndexOf(s String)</code> RETURNING <i>result</i> Numeric	Returns a Numeric value representing the index within the current String of the last occurrence of <i>s</i> , searching backward.
<code>lastIndexOf(s String , index Numeric)</code> RETURNING <i>result</i> Numeric	Returns a Numeric value representing the index within the current String of the last occurrence of <i>s</i> , searching backward. starting at the specified position <i>index</i> .
<code>length()</code> RETURNING <i>result</i> Numeric	Returns a Numeric value representing the length of the current String.
<code>matches(s String)</code> RETURNING <i>result</i> Boolean	Returns a Boolean value specifying whether the current String matches the regular expressions.
<code>replace(old String , new String)</code>	Replaces <i>old</i> in the current String with <i>new</i> .
<code>replaceAll(old String , new String)</code>	Replaces every occurrence of <i>old</i> in the current String with <i>new</i> .
<code>replaceFirst(old String , new String)</code>	Replaces every occurrence of <i>old</i> in the current String with <i>new</i> .
<code>startsWith(s String)</code>	Returns a Boolean value specifying whether the current String begins with <i>s</i> .

Name	Description
RETURNING <i>result</i> Boolean	
startsWith(<i>s</i> String , <i>index</i> Numeric) RETURNING <i>result</i> Boolean	Returns a Boolean value specifying whether the substring in the current String beginning at the position <i>index</i> contains <i>s</i> . The first byte of a string is at position 0.
substring(<i>sindex</i> Numeric , <i>eindex</i> Numeric) RETURNING <i>result</i> String	Returns the substring starting at byte position <i>sindex</i> and ending at the character at <i>eindex</i> -1. The first byte of a string is at position 0.
substring(<i>index</i> Numeric) RETURNING <i>result</i> String	Returns the substring starting at the position <i>index</i> in the current String. The first byte of a string is at position 0.
toLowerCase()	Converts the current String to lowercase.
toString()	Converts the current Numeric value to a String.
toUpperCase()	Converts the current String to uppercase.
translate() RETURNING <i>result</i> String	Uses the current String as the key for a lookup in Genero localization files; returns any entry found, otherwise returns the current String itself.
trim()	Removes white space characters from the beginning and end of the current String.
trimCompress()	Removes white space characters at the beginning and end of the current String, as well as any contained white space.
trimLeft()	Removes white space characters from the beginning of the current String.
trimRight()	Removes white space characters from the end of the current String.
urlencode() RETURNING <i>result</i> String	Returns a URL encoding of the current String.

Usage and Examples

With RTL classes, it is not possible to create and subclass objects. The `new` keyword is not supported.

All literal String values in an expression must be delimited by double quotes.

All the methods require an object instance. When you invoke the method, it is prefixed with the object instance name and the "." character. You can get an object instance by referencing a 4GL variable or by calling a method on another object. The object can be a literal value, for example:

```
"Test".length()
```

Numeric data items in WordBoxes and WordWrapBoxes

If you enter an expression for the text property of a WordBox or WordWrapBox, the value must be a String. You can use the **toString()** function in your expressions to convert numbers to Strings. When you drag a Numeric data item onto the Report Design Window, it is automatically placed in a WordBox element, and an expression for the **text** property is created to convert it to a String.

For example:

```
order_line.unitprice.toString()
```

The indexes of a String (example subString)

When specifying the character position (index) of a string, the first character value is at position 0.

For example, when using the **subString** function, the substring begins at the specified *startIndex* and ends at the character at *endIndex* -1. The length of the string is *endIndex* minus *startIndex*.

```
order_line.billState.subString(1,5)
```

If the value of the String billState is "smiles" (indexes 012345), the substring returned is "mile", and the length of the string is 5 minus 1 = 4.

Concatenating Strings

Use the + operator to concatenate strings. For example:

```
("Total: "+" "+order_line.totalorderprice).toString()
```

This expression returns the current value of totalorderprice as part of a String value:

```
Total: 12.95
```

Upgrading Genero Report Designer

These topics talk about what steps you need to take to upgrade to the next release of Genero Report Designer, and allows you to identify which features were added for a specific version.

- [What's new in Genero Report Designer, v 3.00](#) on page 656
- [What's new in Genero Report Designer, v 2.50](#) on page 838

Upgrading reports from prior versions

The `gsreport` command line utility updates report design documents (.4rp) from previous versions to the current version.

Syntax

```
gsreport [OPTIONS] filename [filename [...]]
```

Table 225: gsreport options

Option	Description
-h	Show help.
-V	Display program name and version.
-c	Convert old format 4rp files to new format
-encoding <i>ENCODING</i>	Sets the encoding to <i>ENCODING</i> . Note: When you use the -h option to display command help, the default encoding for your current environment displays.
-translate <i>LOCALE</i>	Sets the translation file to <i>LOCALE</i> .

Usage

This command line utility accepts a list of files, separated by a space. To include all report files in a directory, use the wildcard symbol (*).

Invoked without any options, the program performs a dry run that reports issues but does not modify the files.

Use option “-c” to convert files to the current version.

Warning: The tool does not back up the files.

If the data schema file (.rdd or .xsd) associated with the report design document is not encoded in system encoding, the -encoding option should be used to specify the encoding.

If error messages and warnings need to be displayed in a language different from the system language, the “-translate” option can be use to specify an alternate language.

What's new in Genero Report Designer, v 3.00

This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication.

Table 226: Genero Report Designer, Version 3.00

Overview	Reference
Genero Report Designer provides a LastPageFooter section property.	See section (Section) on page 754.
Support of Intelligent Mail bar code type.	See intelligent-mail on page 795.
New smartParse bar code property for bar code Code-128. When enabled, this allows you to enter the bar code value, and the internal code will be computed for you resulting in the shortest visual representation.	See smartParse (Smart Parse) on page 755 and code-128 on page 769.
New gs1* bar code aliases.	See Bar Code type listing on page 765.

What's new in Genero Report Designer, v 2.50

This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication.

Table 227: Genero Report Designer

Overview	Reference
Tables support.	See Working with tables on page 691
Pivot tables support.	See Working with Pivot Tables on page 703.
Distributed mode. Allows the report engine to be started as a daemon to which Genero applications can connect to process the reports, allowing for vastly faster processing for short documents and improved scalability.	See Distributed Mode on page 859.
PDF enhancements. Improved PDF output, to include better memory consumption, use of the PDF referencing mechanism to improve Page M of N processing, share recurring images and CID keyed fonts support.	No further reference.
Null value support.	See The String Class on page 834 and The Numeric Class on page 821, Conditional Expressions .
Improved trigger updates. Algorithm improved to remove the need for frequent manual adjustments for each change within the data schema (rdd) file.	See Triggers on page 669.
Runtime localization. Report can now be localized independent of the language settings of the application.	See Change localization settings at runtime on page 598 and fgl_report_configureLocalization on page 616.
QR code barcode support.	See qr-code on page 796.
Display position of footers. Layout nodes designated as footers display at the bottom of the Mini Page, providing a WYSIWYG experience for the report designer.	See Page headers and footers on page 670.
Element creation by context. Create elements based on the document context in the report design. The object type created for a field is determined by the location in the document.	See Adding data values and captions on page 667.
Splitting of oversized elements across pages to prevent overflow.	See splitOversizedItem (Split Oversized Items) on page 755.
Rotation of items. The transformTransparently property changes the effect of the properties layoutDirection and swapX. When set, the transformation extends to the entire fragment so that entire documents can be rotated.	See transformTransparently (Transform transparently) on page 758.
Backside printing support.	See Backside printing on page 691.
Chart sorting. For MapCharts and CategoryCharts, the sortBy property allows you to specify how the data is sorted: alphabetic, numeric, or by order of declaration of the chart items. The sortAscending property allows you to sort in ascending or descending order.	See sortBy (Sort By) on page 755 and sortAscending (Sort Ascending) on page 755.
Fallback image support when the requested image for an Image Box is not found.	See Image Box on page 727.
Edit triggers with a Repeat selected items on menu option in the context menu in the Report Structure view, allowing you to select a trigger to be the parent of a document node.	See Place a trigger within the report structure on page 669.

Overview	Reference
Class property added for report elements.	See class (Class) on page 740.
Display and modify the sizing policy of containers.	See Modify the sizing policy of containers on page 664.
The fidelity property has been added to business charts and the pivot table, applied only when the object in question is drawn as a table.	See Business Graphs on page 731.
The layout direction of a parent container is highlighted in the Genero Report Designer by the addition of a dashed, slowly moving, U-shaped yellow border.	See layoutDirection (Layout Direction) on page 749.
Preference added to control the appearance of RTL expressions in the document view.	See Customize Report Designer: preferences on page 806.
Added options to facilitate the mass generation of images that are sized by their content (e.g. for web sites).	See fgl_report_setImageUsePageNamesAsFileNames on page 631 and fgl_report_setImageShrinkImagesToPageContent on page 631.

Report templates

A report template defines the layout of a professionally-designed report that you can use to quickly create an initial report design.

When you have multiple reports with the same structure, but with different data sources, you can use report templates to provide a consistent look-and-feel for all similar reports.

A template is a `.4rt` file. It is a general report design that is not tied to a specific data source. It is a graphical design of a report, and you associate data with data placeholders that have been set in the graphical design. You decide which data to inject by specifying the data source, when you select the data schema to use.

Once the data source is selected, two things need to be mapped: the structure of the data with the structure of the template, and the fields from the schema to the data placeholders in the template.

When creating a report, you can choose from the existing list of templates, or you can create your own report template.

Create a report from an existing template

In this procedure, the New Report from Template wizard is used to create a report design document (`.4rp`) from an existing template.

The New Report from Template wizard results in a report design document (`.4rp`) being created. Once created, the report design document is a stand-alone document and no longer has any connection to the template that created it. Any changes made to the report template (`.4rt`) have no effect on existing report designs (`.4rp`).

1. Select **File >> New, Report From Template**.

A list of templates appear. You can refine the types of templates that appear in the list by selecting from the **Filters** drop-down list. See [Template filters in the Designer Wizard](#) on page 846. Click on a template icon to view a sample of the report that can be created from that template.

2. Select a template and click **OK**.

The **New Report From Template** wizard opens.

3. In the **Schema Association** page, you provide information about the data source.

For complete details on this step, see [Schema Association page](#) on page 841.

- a) In the **Schema Location** field, select a data schema (.xsd).
Once selected, a list of available classes appears in the Schema Root combo-box. The Record1 appears in the rows box.
- b) In the Schema Root combobox, select the schema root.
- c) Complete the Repetitions section, mapping the template repetitions to the schema repetitions.
- d) Click **Next >**.

4. In the **Add Fields** page, select the fields to display in a report object designed to accept a variable number of fields.

Refer to the report diagram on the right-hand side of the wizard, the object will be highlighted. A new **Add Fields** page displays for each variable-field object contained on your report. For complete details on this step, see [Add Fields page](#) on page 842.

5. On the **Variables** page, provide values for variables in the template.

The **Placeholders** section provides you with the names of the various variables you can modify. Click on a variable, and the report image highlights the area or object that is affected by the variable.

For complete details on this step, see [Variables page](#) on page 844.

6. Select **Finish**.

The report is created as an independent .4rp file. It is no longer associated with the template. It is a report design definition, and can now be treated as a standard report design document (.4rp) file.

Schema Association page

The **Schema Association** page in the **New Report From Template** wizard lets you select your schema and associate schema repetitions to template repetitions.

There are two sections on the **Schema Association** page.

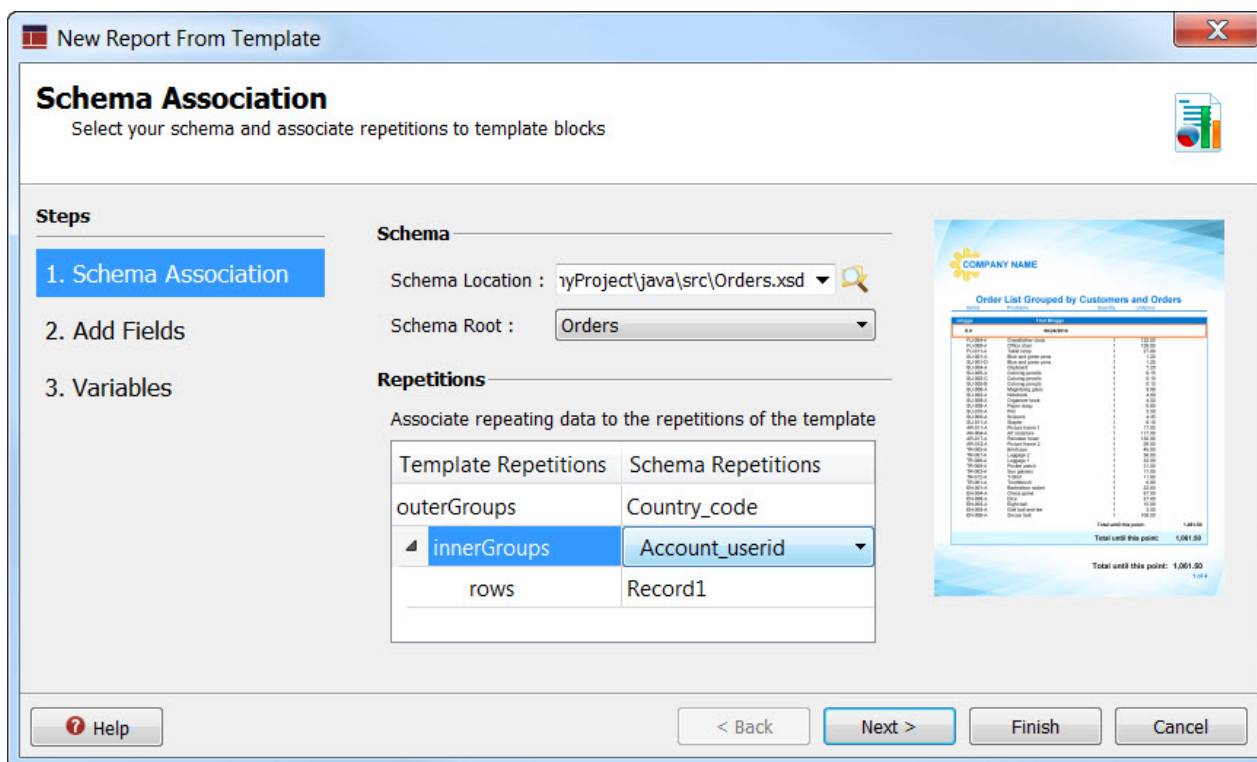


Figure 384: Schema Association

Schema

In the **Schema Location** field, you select the name of the data schema file to use. Selecting this file populates both the **Schema Root** combobox and **Repetitions** section with suggested default values.

The **Schema Root** field, select the root of the data model. If you have a master and details listed, select the master. If the data schema was generated with the Business Application Modeler, there will only be one schema root.

Repetitions

A report template has repetitions. You can repeat the whole document, you can repeat sections of the report, and you can repeat each row. The sections of the report that repeat are listed under the **Template Repetitions** column. When you click on a row, the associated repetition is highlighted in the report sample.

A data schema has repetitions. These repetitions represent the hierarchy of data, as defined in the `ORDER BY` clause in the query used to extract the data. Once added to a report design document, these repetitions are identified by [report triggers](#). When you click on a cell in the **Schema Repetitions** column, a combobox appears with a list of possible schema repetition fields to select from.

In completing this section, map the template repetitions to the schema repetitions. If the data schema has groups defined, the wizard attempts the mapping for you.

To change a group setting, click in the **Schema Repetitions** cell for that group. A combobox appears, allowing you to select one of the other grouping fields defined in the data schema. The section of the report that relates to this group is highlighted in the report sample.

The last row of the repetitions box is for the individual data rows themselves; the rows that will appear in the listing. For most reports, this is going to be the record defined in the data schema.

Add Fields page

The **Add Fields** page in the **New Report From Template** wizard lets you select fields for a report object that is designed to accept a variable number of fields.

This page displays for each report placeholder that can accept a variable number of fields. As such, it may not appear at all, or it may appear multiple times. The report image on the right-hand side of the wizard highlights the area of the report that is in focus when using this page.

There is only one section in the **Add Fields** page, the **Select fields** section.

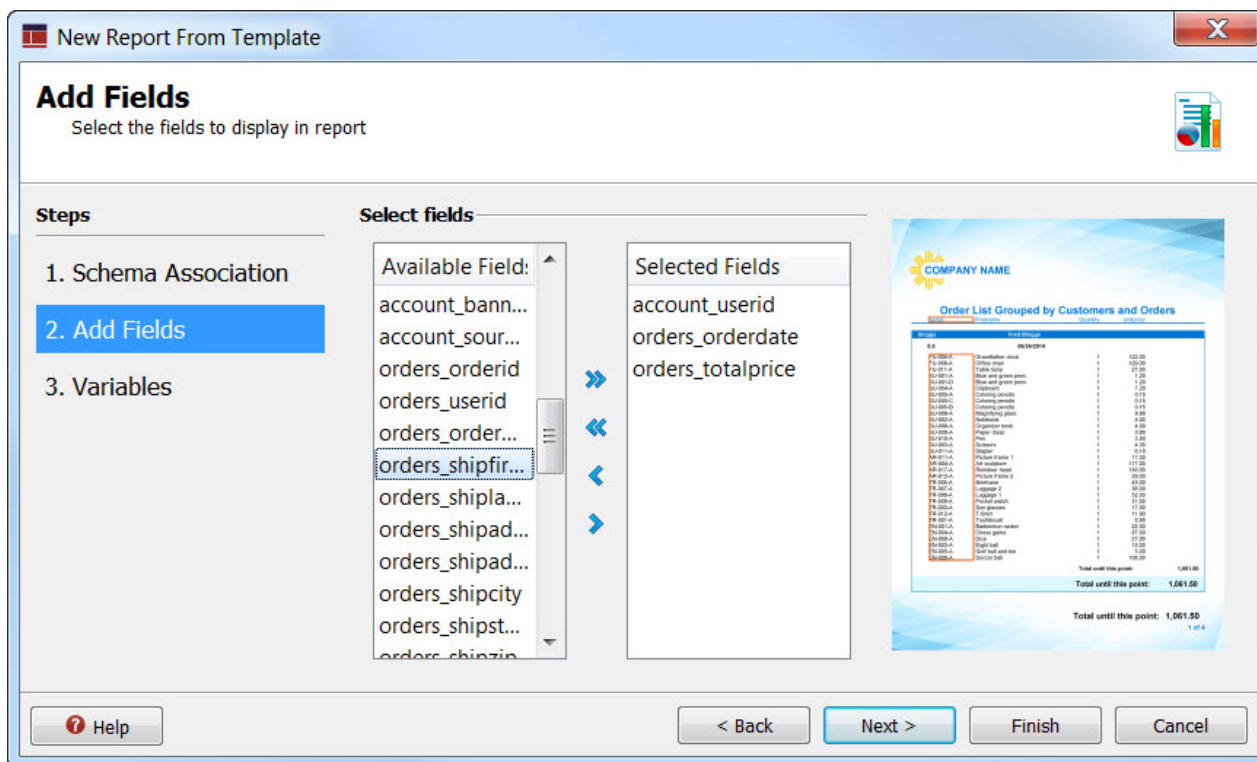


Figure 385: Add Fields

Select the fields to use for the highlighted placeholder. While you can click the double-arrow icon to include all fields in the placeholder, it is typical to pick a subset of the fields to populate the placeholder.

The order of the columns in the list determines the order they appear in the row.

Aggregate fields

As a report designer, you have received a data schema from the reporting application developer. The developer may have created this application using the Business Application Modeler. One of the options available to the developer is to specify that a field be aggregated: give me the sum of the price of all orders, or give me a count of the number of orders.

In the **Select fields** list, these aggregates appear for each level of the grouping, as well as for the overall report.

For example, each order has a field called `totalprice`, that gives the total price for that order. The total price is something that is typical to summarize in a report, so the developer will specify that the SUM should be provided for the `totalprice` field. The data is grouped by account and by country, which means we are given fields for the sum of all orders for an account or the sum of all orders for a country.

The names generated for these fields are shown in [Figure 386: Aggregate fields](#) on page 844.

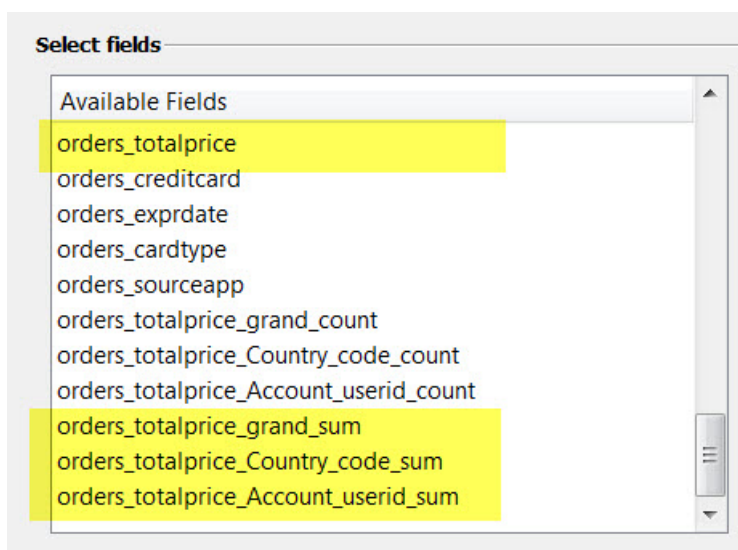


Figure 386: Aggregate fields

The non-summarized field in this example is `orders_totalprice`. When the designer specified that a summary be kept, the additional fields added were:

- `order_totalprice_Account_userid_sum` to hold the sum of the totalprice for the current account.
- `order_totalprice_Country_code_sum` to hold the sum of the totalprice for the current country.
- `order_totalprice_grand_sum` to hold the running total of the totalprice for all the rows.

Further examination of the fields in our example also show the aggregate fields created for the count aggregate.

Variables page

The **Variables** page in the **New Report From Template** wizard lets you provide values for the variables defined in the template.

There is only one section in the **Variables** page, the **Placeholders** section.

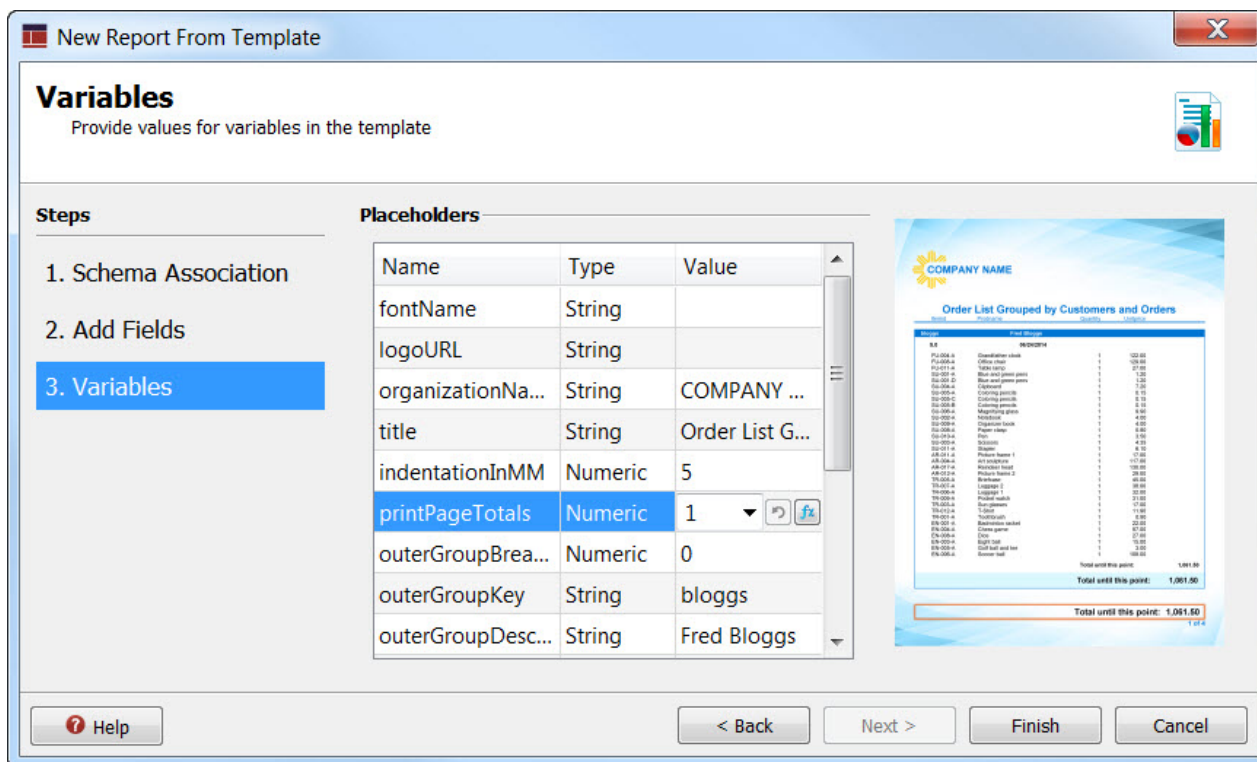


Figure 387: Variables page

In the **Placeholders** section, a table lists all of the template variables. Click on the row containing a variable to see its location in the report; as you select a variable, the section of the report that relates to the variable highlighted in the report sample.

Each variable can be set to a single value. There are three types of values that can be applied:

- A value can be static text. To enter a fixed value, type the value directly into the cell.
- A value can be a field name from your data source. Click on a cell to activate the combobox and select a field from the list.
- A value can be an expression. Click on the expression icon (fx) to launch the expression editor.

Tip: An expression can be comprised of multiple fields. For example, you may wish to put a customer's first and last name in a field. You would use an expression to concatenate the fields, with proper spacing.

The types of variables will depend on the template; a template designer has complete freedom to define the variables necessary for the template being developed. Here are some examples of types of variables:

- Variables can hold strings. Common variables of this type are report titles.
- Variables can display the content of a field or fields. Common variables of this type would be including grouping data in sub-sections of a report.
- Variables can be boolean flags. You can have a variable that includes a section to print page totals, based on whether you specify 1 (true) or 0 (false). For example, see printPageTotals in [Figure 387: Variables page](#) on page 845

These are just a few examples of the types of variables that one may have added to a report template. As a user of a template, ensure you know what each variable does, and what types of values are expected.

Template filters in the Designer Wizard

The templates provided with Genero Report Designer offer a wide range of options to cater for the many types of report that you can create. When viewing New Reports from Templates in the Designer Wizard, you can refine the templates that appear in the list by selecting options from the **Filters** drop-down list.

To assist you in searching for the template most suited to your report type, you can refine the type of templates that appear in the list by selecting from the list of filters provided.

[Table 228: List of default filters](#) on page 846 provides a list of the default filters and a description of each filter.

Table 228: List of default filters

Filter	Description	Sample template
Batcheable	Templates that can be used create a single document/print job that contains multiple documents (e.g. you create a single print job that contains invoices from different customers). The difference is that if you create individual documents, other user may submit a print request that will be printed in the middle of the batch.	Invoice (PULSE).
Correspondence	Templates for reports that can be used correspondence purposes, typically containing an address.	Simple Invoice (PULSE).
Grand Total	Templates that can be used to print grand totals so that a running total is calculated with the data.	Group List with Totals (PULSE).
Group Headers	Templates that contain headers for groups of data.	Group List with Totals (PULSE).
Group Totals	Templates that can be used to print group totals for the groups available from the data.	Group List with Totals (PULSE).
Invoice	Templates that create an invoice document	DIN 5008 Invoice (PULSE)
List	Templates that create a list report.	Form List (PULSE).
PULSE Theme	Templates that use the PULSE graphical style.	Simple List (PULSE).
Totals Block	Templates that create a summary block on the last page.	Invoice (PULSE).
Two Groups	Templates that requires the data to be grouped by at least two dimensions.	Simple Invoice (PULSE).

Note: When creating a new template, you can add existing filters to your template or create new filters to add to the list. See [Customize the appearance of a new report template in the wizard](#) on page 851.

Template expansion at runtime

Genero Report Writer provides the option to expand a template dynamically at runtime.

You can create a program to add data sources and the associated mapping to a template at runtime. This method uses a report template (.4rt) and design-time APIs to provide the details about the schema, the schema root, the relationships, and the field mappings in order to output the report, bypassing any need for a report design document. While more complex to set up, the advantage is that changes to your template are reflected with each new run of the report.

Genero Report Writer allows you to expand templates at runtime, thereby providing end users with generic reports. Generic reports typically present themselves to end users in the following three phases:

1. The user is prompted for information regarding the template to use and the values to use (variables and placeholders). While all the same options as sophisticated as the template assistant in the report designer are available, typically the following things will be simplified:
 - You can't choose multiple different templates. If choices are given, they are limited to styles that match the data source (e.g. don't offer a grouped list when the data source doesn't have groups).
 - You won't be prompted for difficult placeholders, for example those that require the construction of formulas (e.g. Total expressions), and will only be prompted for simple values (e.g. the Title of the report).
 - You must choose fields from a single list of fields. Therefore, templates that offer more than one field lists will not be used.

Note: Building the dialog in a generic way to avoid hard coding placeholder and field lists requires software that can introspect schema files and .4rt files. A library is provided for that.

2. The information entered by the user is used to expand the template and generate a .4rp file.

Note: Expanding the template can be done in one of two ways:

- By calling an existing library function.
- By invoking the [GenerateReport](#) executable provided with GRE.

3. The data source is run using the generated .4rp file.

Note: The data source is run with the .4rp via the normal runtime API functions.

Create a new report template

In addition to the templates provided, you can create your own report templates and add them to the Designer Wizard.

To create a new report template, first create a report template schema definition (.rsd) to define the expected structure for the data source and then create the report template (.4rt). Any new templates must be added to the Designer Wizard.

Before you begin:

- Create a new directory on your disk (e.g: My Report Templates) where you will store the report template files that you will create in the following procedure.

Create a new report template schema

1. Go to **File > New > Reports** and click on **Report Template Schema Definition(.rsd)**.
A new .rsd file opens.
2. Using the available elements, create a structure for the elements that will be used as placeholders in the report template. See the [Report template schema definition \(.rsd\) file](#) on page 849 topic for more information.
3. Save the file.

Create a new report template

4. Go to **File > New > Reports** and select **Empty Report Template(.4rt)**.
A new .4rt file opens.
5. In the **Data View** tab, click **Open Schema File...**
6. Navigate to the location that you saved your .rsd file in Step 3, select the .rsd file, and click **Open**.
The structure of the elements from the .rsd file that you selected is visible in the **Data View** tab.
7. Save the .4rt.
You have now created a report template that you can design to suit your requirements.

Note: Before you design your report, see [Report template \(.4rt\) file design features](#) on page 850 for more information.

Add the new report template to the wizard

8. Create a new configuration file.

Note: The default configuration file for the Design Wizard is the `createables.conf` file, located in the `$GSTDIR/conf` directory. To create new settings for the template, you must create a new `createables.conf` file, save it in your new directory, and configure the template to use the new `createables.conf` file.

- a) Go to **File > New > Other files** and select **With no Extension**.
 - b) Save the file to your new directory as `createables.conf`.
9. Add the following code to the new `createables.conf` file and edit the label, name, and `directoryPath` accordingly:

```
<?xml version="1.0" encoding="utf-8" ?>
  <Creatables version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xsi:noNamespaceSchemaLocation="http://www.4js.com/ns/gst/3.00/
createables.xsd">
    <Directory index="35" label="<newreportlabel>"
name="<newreportname>"
  icon="document_4rp" directoryPath="<./My Report Template>">
      <DocumentType extension="4rt" icon="document_4rt"
action="RWTemplateWizard"/>
      <DocumentType extension="4rp" icon="document_4rp"
action="RWTemplateWizard"/>
    </Directory>
  </Creatables>
```

10. Configure the new `createables.conf`.

A reporting template is provided in the **Genero Configurations** dialog that can be duplicated for your convenience and then associated with your configuration file.

- a) Go to **Tools > Genero Configurations**.
- b) In the **Environment Sets** list, right-click on **Reporting template 1.0** and select **Duplicate**. A new reporting template is created at the bottom of the **Environment Sets** list and is selected by default.
- c) Click on the new reporting template. The `GSTSETUPDIR` directory is listed in the **Environment Variables** section.
- d) Double-click the `GSTSETUPDIR` entry and change the **Value** to the location of your new template directory.
- e) Click **OK** to close the dialogs.
- f) In the **Information** dialog, click **Reload Session**.

When you have completed the above procedure, go to **File > New** to open the wizard and view your new report template in the list.

Report template schema definition (.rsd) file

The report template schema definition (.rsd) file is used to manually add the structure of the fields and groups to a report template (.4rt).

The .rsd file is a high level data schema file, unique to templates, that is used to create the XML structure of the elements that will be included in the report template as placeholders. When creating a report from a template, the .rsd placeholders are populated with the data of the report design document provided by the data schema, or XML Schema definition (.xsd) file.

When adding the syntax to the .rsd, you must first add a `rootElementName`, and then structure the file with the following available elements:

Table 229: Available elements in the .rsd document

Element	Required syntax	Description	Reference
Field	<ul style="list-style-type: none"> name=" " type=" " sampleValue=" " (optional) 	The field element is used to add the variables that you want to include in the report template as placeholders.	See the Variables page on page 844 topic for more information.
Trigger	name=" "	The Trigger element represents the groups that you want to add to the report template as template repetitions, which can be mapped to the repetitions in the data schema.	See the Schema Association page on page 841 topic for more information.
TemplateFieldsTrigger	groupName=" "	The TemplateFieldsTrigger element is added for fields where real field data is expected to be injected. When creating a report from a template and the template is expanded, the option to choose from the available fields in the data schema .xsd is provided.	<ul style="list-style-type: none"> See the Schema Association page on page 841 for more information about associating the data schema. See the Add Fields page on page 842 topic for more information about selecting real data fields.

See the following sample taken from the .rsd file for the **DIN 5008 Invoice (PULSE)** template, provided with Genero Studio for Genero Report Writer:

```
<?xml version="1.0" encoding="utf-8"?>
<ReportSchema fileVersion="30000" gstVersion="30000"
  rootElementName="model">
  <Field name="showMeasures" type="boolean" sampleValue="1"/>
  <Field name="fontName" type="string"/>
  <Field name="logoURL" type="string"/>
  ...
  <Trigger name="outerGroups">
    <Trigger name="innerGroups">
      <TemplateFieldsTrigger groupName="fields"/>
    </Trigger>
  </Trigger>
</ReportSchema>
```

```

    <TemplateFieldsTrigger groupName="fields" />
    <Trigger name="rows">
      <TemplateFieldsTrigger groupName="fields" />
    </Trigger>
  </Trigger>
</ReportSchema>

```

Examine the various template schema files in `$GREDIR/templates` to better understand the different ways that a template schema file can be written.

Report template (.4rt) file design features

The report template (.4rt) file is the template file and has some unique design features over standard report files (.4rp).

When you create a report template (.4rt) file and want to add design elements, the process of designing the template is almost identical to [designing a report](#), except for the following unique features:

The Report Structure: TemplateFieldsTrigger

When you have created the .rsd file and have added the `TemplateFieldsTrigger` elements to add placeholders for where you want to inject real data, the color of the template fields trigger is displayed in the Report Structure view. The template field trigger is blue to distinguish it from the other triggers. See the [Triggers](#) on page 669 topic for more information.

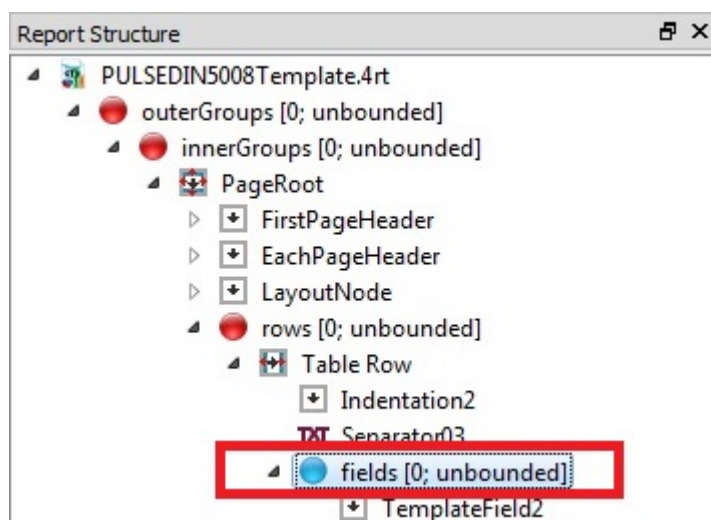
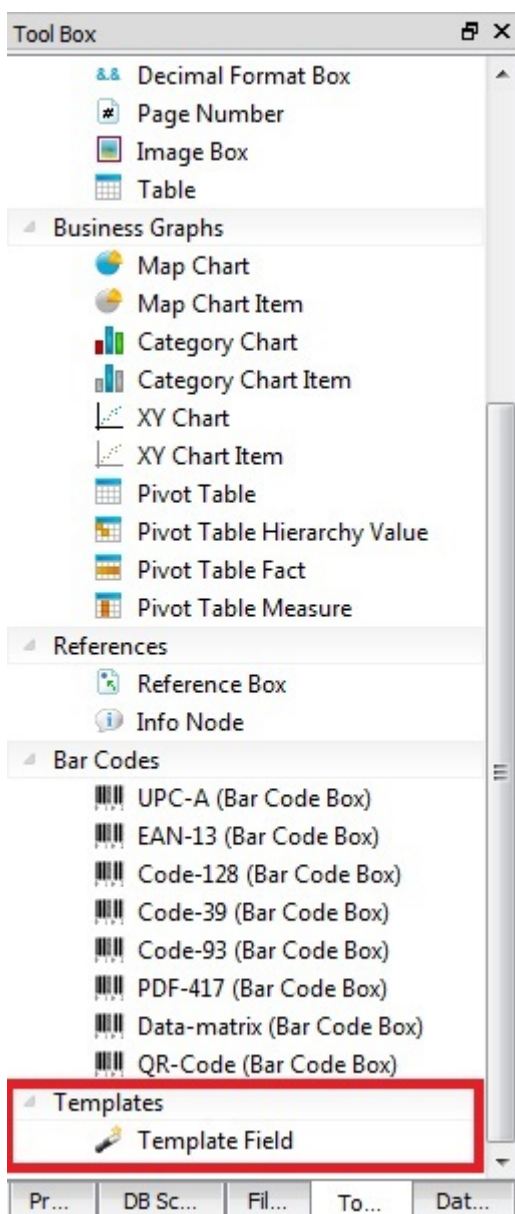


Figure 388: The temple fields trigger in the Report Structure view

The Tool Box view- Template field element

When designing a report template, the Tool Box view has an extra Templates section that includes a design element called the **Template Field**.

Figure 389: The Template Field element in the Tool Box view



You can add the template field element to elements in the report that will be expanded with real data when you use the template to create a report, and should be placed within the template fields trigger groups in the Report Structure view.

Customize the appearance of a new report template in the wizard

You can add text and images to customize the appearance of your new template in the Wizard.

Templates in the Designer Wizard include preview images, a label and description, and are organized into different categories by filters. When you have created a new template and added it to the design wizard, you can customize the appearance of your new template in the wizard.

Template label, description, and filters

Templates are categorized by filters. You can view the list of filters by going to **File > New...**, selecting **Report from Template**, and clicking on the **Filters** drop-down list.

Each template has a label, a description, and is filtered by the tags that are included in the associated properties file (.4rt.prop). For more information about the list of existing filters, see [Template filters in the Designer Wizard](#) on page 846.

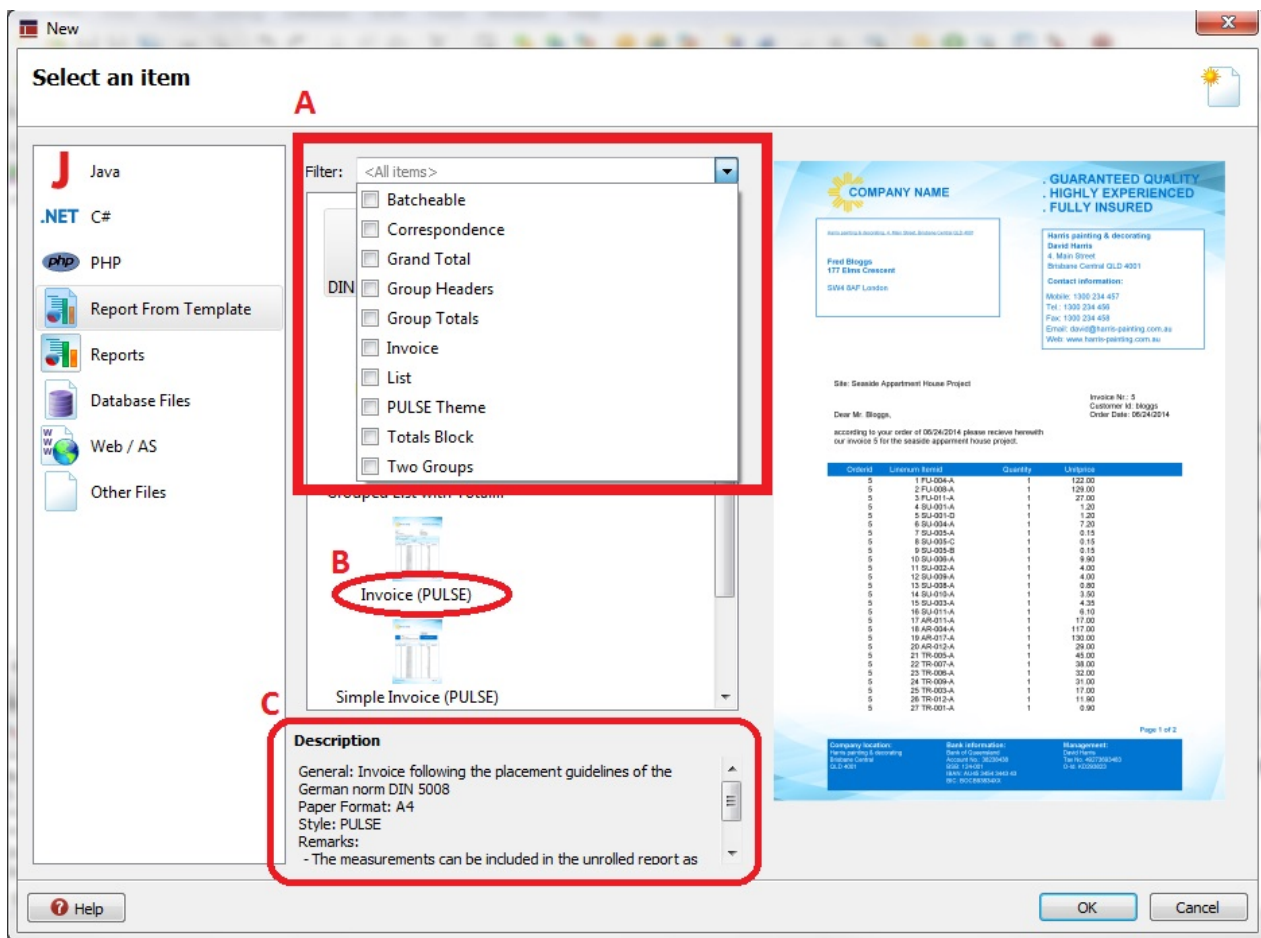


Figure 390: Report from Template wizard: A = Filters, B = Label, C = Description

See the following example of the syntax in the PULSEDIN5008Template.4rt.prop:

```
tags: Correspondence, Invoice, Batcheable, Two Groups, PULSE Theme
label:DIN 5008 Invoice (PULSE)
description: General: Invoice following the placement guidelines of the
  German norm DIN 5008\nPaper Format: A4\nStyle: PULSE\nRemarks:\n - The
  measurements can be included in the unrolled report as an editing aid\n
  - The report is batcheable by the outer group
```

Template images

In the Report From Template page in the Designer Wizard, each template has a thumbnail image in the list and a large version of the same image displayed on the right of the page. This image is a screen capture of the report and is stored in the `$ProjectDir/gre/templates` directory as `<TEMPLATENAME>.4rt.png`.

When you have selected a template in the Designer Wizard and the New Report from Template Wizard opens, images are displayed in each of the three pages; the [Schema Association page](#), the [Add Fields page](#), and the [Variables page](#), that are designed to call-out the specific section of the report that is relevant to the current selection in the form.

The sample template images are stored in the `$ProjectDir/gre/templates/<TEMPLATENAME>` directories and adhere to the following naming conventions:

- Schema association images: `<PLACEHOLDERNAME>_placeholder.png`
- Add fields images: `field_field<NUMBER>.png`

- Variables images: <TRIGGERNAME>_trigger.png

Create a report template from an existing report

You have a report that you like. You can use that report as a basis for a new report template.

You have an existing report, and you want to use it as the basis for a new report template. When you have finished, you must add the new templates must be added to the Designer Wizard.

Before you begin:

- Have a new directory on your disk (e.g: My Report Templates) where you will store the report template files that you create in the following procedure. If you have not set up a template directory, see [Set the report template directory](#) on page 854.
- Have an existing report.

1. Create a copy of your report design (.4rp) and save it in your report templates folder.

2. Rename the copy to a different name, and with a .4rt extension.

It is recommended that you not have a .4rp and .4rt sharing the same name in a template folder. If your report is names SalesList.4rp, you would not simply name it as SalesList.4rt. You would modify the name in some manner; for example, you can add the word Template to the name and save it as SalesListTemplate.4rt.

3. Open the .4rt file in Report Designer and examine the report. In the next step, you must have to identify which fields are to be variable placeholders and which parts are to be template fields.

4. Create a template data schema (.rds).

Rather than starting with an empty file, it is recommended that you take an existing report data schema (.rsd) file from the templates provided during the installation and modify it to meet your report needs. You can find template data schemas (.rds) in \$GREDIR/templates.

5. Go back to your .4rt file, to the Data View tab, and change the schema to the schema you created in the previous step.

You are not going to use the actual data fields anymore, you are going to use the fields from your template schema. In the Design View, the places where you had data fields (from the original report) are now marked as errors.

6. For the variable placeholders, double-click on each error to bring up the **Edit Expression** dialog. Replace the value with one of the names from the template schema file.

7. For the row of fields, such as those contained within a Table Row container, you must replace the fields with a Template Field placeholder.

a) Delete the existing fields.

b) Go to Tool Box.

c) Drop the Template Field onto the Table Row (or similar container).

d) In the Template Field section in the Properties view, the Name, Type, Size, Title and Role properties should be preset.

When you drag something from the Data View into the Document View, it creates an object based on the context of where it is being placed. These five properties provide the information needed to create the correct object. This field is repeated as many times as needed.

8. If you have column headers, you will also have to replace the column headers with a Template Field.

9. In the **Report Structure**, organize the TemplateField objects to sit under the fields triggers.

Click on a TemplateField with the right mouse button, select **Repeat selected items on**, then select the template fields trigger.

At this point, the template is finished.

10. Create a copy of a .prop file, and modify the contents of the file for your new template. Name the copy using the same name as the template file.

11. Create a template image (.4rt.png) to serve as the image displayed in the Report From Template selection list. Save it using the same name as the template file.

12. Create an empty image directory for the new template, using the same name as the template file.

Eventually, you will want to provide an image for each of the placeholders. This can be done at a later time.

13. Select **Tools >> Specific setup >> Reload**.

The new template is available for use.

Modify an existing report template

You can modify an existing report template and add them to the Designer Wizard.

To modify an existing report template, you need to create copies of the existing template files and then modify the copies. When finished, the modified templates must be added to the Designer Wizard.

1. Create a new directory on your disk (e.g: `My Report Templates`) to hold your customized report template files.
2. Locate the files for the existing report template, and move them into your new directory.

These files include:

- `*.4rt` - the existing report template file.
 - `*.4rt.png` - the existing image used for the report template file in the new report from template wizard.
 - `*.4rt.prop` - the existing configuration file used to categorize the report template.
 - `*.rsd` - the report schema design file used by the report template.
 - the template-specific sub-directory, containing the many images used by pages within the template wizards.
3. Rename the files and sub-directory to use a unique name. All of the files (with the exception of the `.rsd` file) and the image directory should share the same name.
 4. Make changes to the `*.4rt` file in Report Designer.
 5. Update the `*.4rt.prop` file to specify your template name, description, and filtering tags.
See [Customize the appearance of a new report template in the wizard](#) on page 851.
 6. Update the `creatables.conf` file in your user-based `AppData/Roaming/FourJs/GRW/tpl/reporting1.1` directory and add details about the directory.
See [Set the report template directory](#) on page 854 for more details.
 7. Update the image files, as necessary. See the section on template images in [Customize the appearance of a new report template in the wizard](#) on page 851.

Set the report template directory

When you create a new report template directory, you need to complete some simple configuration before the templates in that directory are seen by Genero Studio.

You have created a directory to hold your custom template files. You must update the `creatables.conf` configuration file to make Genero Studio aware of this directory. Once added to the configuration, all templates in the directory appear in the **Report From Template** selection list.

The initial template directory is defined in the `creatables.conf` file provided in the Genero Studio installation directory (at `$GSTDIR/conf`). It is recommended that you add additional directories in your user-specific `creatables.conf` file.

Before you begin:

- Know the full path to the new template directory.
1. Update the `creatables.conf` file in the user's `AppData/Roaming/FourJs/GRW/tpl/reporting1.1` directory and add details about the directory.

Use this example as a guide:

```
<DocumentDirectory index="35" label="Report From Template"
name="RWReportFromTemplate" icon="document_4rp" directoryPath="D:/
myTemplates">
  <DocumentType extension="4rt" icon="document_4rt"
action="RWTemplateWizard"/>
  <DocumentType extension="4rp" icon="document_4rp"
action="RWTemplateWizard"/>
</DocumentDirectory>
```

2. Select **Tools >> Specific Setup >> Reload**.

3. Click **Reload session**.

The templates within the specified directory appear in the **Report From Template** list.

GenerateReport command options

The `GenerateReport` command creates report design files (.4rp) based on a predefined template and schema.

Table 230: GenerateReport options

Option	Description
-help or -h	Displays a usage text and then exits.
-schemaFileName Important: Mandatory option.	Specifies an XML schema file (.xsd) describing the data source of the report.
-rootElementName Important: Mandatory option.	Specifies the expected document root in the XML schema file. For example, if the schema specifies the elements "invoice" and "invoice-batch", then <code>rootElementName</code> would be set to "invoice" if the report will be run against a source that produces documents whose root element is of type "invoice".
-triggerMapping Important: Mandatory option.	Specifies the mapping between the element names in the XML schema file and the trigger names in the design template. The map syntax is as follows: <pre>map: map-item (',' map-item)* map-item: '{' element-name ',' trigger-name '}'</pre> As an example, consider a report template designed against the schema defined by <code>SimpleListTemplate.rsd</code> . Note: Report template schema definition (.rsd) files are located in <code>GREDIR/templates</code> . The schema defines the mappable triggers "outerGroups", "innerGroups" and "rows", where each is a descendant of its predecessor. If the input schema defines the elements

Option	Description
	<p>"ProductGroups", "Areas", "Orders" and "Items", then the following are valid maps:</p> <pre>{ProductGroups,outerGroups},{Areas,innerGroups}, {Items,rows}</pre> <pre>{ProductGroups,outerGroups},{Orders,innerGroups}, {Items,rows}</pre> <pre>{ProductGroups,outerGroups},{Orders,rows}</pre> <pre>{ProductGroups,rows}</pre> <pre>{Orders,rows}</pre> <p>The following example mappings are invalid because they violate the ancestry:</p> <pre>{ProductGroups,innerGroup},{Areas,outerGroups}, {Items,rows}</pre> <pre>{Orders,outerGroups},{ProductGroups,rows}</pre>
-placeholderMapping	<p>Specifies the mapping between fields names in the design template and expressions of the same type that may be composed using fields from the XML schema file. Specifying this value is mandatory if the template contains references to fields.</p> <p>Note: All placeholder values in placeholderMapping can either be constant values or RTL expressions enclosed in curly braces.</p> <p>The map syntax is as follows:</p> <pre>StringMap: MapEntry (',' MapEntry)* MapEntry: '{' Key ',' Value? '}' Key: IdentifierStartChar IdentifierFollowChar* Value: '"' StringToken* '"' StringToken: [^"\\] EscapedQuote EscapedBackslash ExtraEscapes EscapedQuote: '\\ ' "' EscapedBackslash: '\\ ' \' ExtraEscapes: '\\ ' '\n' '\\ ' '\r' '\\ ' '\t'</pre> <p>This means that encoders need to perform the following operations on all characters in the input strings:</p> <ul style="list-style-type: none"> • Replace '\\' by '\\ ' \' • Replace '"' by '\\ ' '"' • Replace '\n' by '\\ ' '\n' • Replace '\r' by '\\ ' '\r'

Option	Description
	<ul style="list-style-type: none"> • Replace '\t' by '\ ' 't' <p>As an example, consider a report template that contains:</p> <ul style="list-style-type: none"> • the string field "groupTitle", mapped to the RTL expression "Customer: "+orderline.orders.user_id" • the field "reportTitle", mapped to the constant string "Customer list" • the placeholder "optionalSubtitle", set to null <p>In this example, the placeholder mapping would be:</p> <pre>-placeholderMapping {groupTitle, "{{\"Customer: \"+orderline.orders.user_id}}"}, {reportTitle, "Customer list"}, {optionalSubtitle, }</pre> <p>This assigns the RTL expression '"Customer: "+orderline.orders.user_id' to the placeholder "groupTitle", the constant string "Customer list" to the placeholder "reportTitle", and the value null to the placeholder "optionalSubtitle".</p> <p>Note: For clarity, no quoting was done to protect the string against shell expansion.</p>
-templateFileName	<p>Specifies the name of the template (a '.4rp' or '.4rt' file) used as the base. If this parameter is not specified, then a default list template is used. The default template is designed against the schema of the SimpleListTemplate.rsd.</p> <p>Note: Report template schema definition (.rsd) files are located in <i>GREDIR/templates</i>.</p>
-fieldNamePatterns	<p>Specifies a selection of fields from the XML schema file that are to be used in the resulting report.</p> <p>The expected syntax is a comma-separated list of field name patterns, which may contain the wildcard characters "*" and "?". The expression can be prefixed with an optional name followed by a colon. This name denotes a specific field trigger to cater to templates with multiple field lists.</p> <p>As an example, consider a report that has the field triggers "outerGroupFields" and "rowFields". We would like to see the fields product_id and product_description on the group, and all fields from the record order_details in the rows. We would specify two fieldNamePatterns as follows:</p> <pre>-fieldNamePatterns outerGroupFields:product_id,product_description -fieldNamePatterns rowFields:order_details.*</pre>
-outputFileName	<p>Specifies the name of the resulting .4rp file.</p> <p>Important: Mandatory option.</p>
-debuglevel <i>level</i>	<p>Sets the debug level to the specified integer level. The debug level controls the level of verbosity of GRE components during execution.</p>

Option	Description
	Higher values increase verbosity. By default, the value is set to 0 (no debugging output).
-stdin	Instructs the program to read the command line arguments from stdin. The list of arguments needs to be terminated by an empty line. In this case, all other regular command line arguments are ignored.

Report Writer Deployment and Customization

The Genero Report Engine (GRE) and Genero Report Viewer for HTML5 have limited configuration and customization options.

- [GRE environment variables](#) on page 858
- [Genero Report Viewer for HTML5 customization](#) on page 859
- [Distributed Mode](#) on page 859

GRE environment variables

These environment variables are relevant for the Genero Report Engine (GRE).

- [GREDIR](#) on page 858
- [GREOUTPUTDIR](#) on page 858
- [GRE_MAX_CONCURRENT_JOBS](#) on page 858

GREDIR

The *GREDIR* environment variable specifies the location of the Genero Report Engine (GRE) installation directory.

GREOUTPUTDIR

The *GREOUTPUTDIR* environment variable specifies the directory where all writing operations that would otherwise be performed in the current working directory of the Genero Report Engine (GRE) will be performed.

The value can be an absolute or a relative file path. If the value is a relative file path, then it is internally prepended with the current working directory of GRE. When set, all writing operations that would otherwise be performed in the current working directory of GRE will be performed in the specified directory.

Examples include relative path specification in the functions `fgl_report_setOutputFileName()`, `fgl_report_configureImageDevice()` (the *filePath* parameter), `fgl_report_setBrowserDocumentDirectory()`, `fgl_report_setBrowserFontDirectory()` and the location of the GRE debug files (`jdebug0.xml`, `jdebug1.xml`, and so on.)

GRE_MAX_CONCURRENT_JOBS

The *GRE_MAX_CONCURRENT_JOBS* environment variable limits the number of worker threads in distributed mode.

By default, the Genero Report Engine runs 25 concurrent threads. As such, by default it can process no more than 25 concurrent jobs. Jobs that started at a point in time when the maximum value has been reached are queued until another job completes. The limit on concurrent threads is to prevent memory exhaustion in times of critical load.

GRE_MAX_CONCURRENT_JOBS allows the modification of the number of concurrent threads. This variable takes an integer.

Genero Report Viewer for HTML5 customization

When you select Browser as the output option, files are created for the report. These files are viewed using the Genero Report Viewer for HTML5.

All files needed to operate the Genero Report Viewer for HTML5 are included in the Genero Report Engine (GRE) package in the directory `$GREDIR/viewer`

Customizing the viewer

The Genero Report Viewer for HTML5 is comprised of the files found within `$GREDIR/viewer`.

Table 231: GRV for HTML5 viewer files and directories

File / Directory	Description
<code>viewer.html</code>	Main HTML file containing the toolbar and document view. Customizable via CSS. Can be replaced by custom version if the same classes, ids and event bindings are used. Loads <code>viewer.js</code> , <code>model.js</code> , <code>browser.js</code> and <code>styles.css</code> .
<code>print.html</code>	Print preview page. Customizable via CSS. Can be replaced by a custom version if the same classes, ids and event bindings are used. Loads <code>print.js</code> , <code>model.js</code> , <code>browser.js</code> and <code>styles.css</code> .
<code>viewer.js</code>	JavaScript file that connects the HTML elements in the viewer (<code>viewer.html</code> by default) with the model (<code>model.js</code>). Can be replaced for the case that a custom HTML viewer is used that does not use the same classes, ids and event bindings as the default viewer.
<code>print.js</code>	JavaScript file that connects the HTML elements in the print previewer (<code>print.html</code> by default) with the model (<code>model.js</code>). Can be replaced for the case that a custom HTML print previewer is used that does not use the same classes, ids and event bindings as the default print previewer.
<code>model.js</code>	A JavaScript object representing a report. Methods exist to query the document (for example, to get the total number of pages), to navigate in the document (for example, to make a certain page the current page) and to register for events (for example, to ask to be notified when a new page is created).
<code>browser.js</code>	A third-party utility for managing browser-specific issues.
<code>styles.css</code>	The default styles used by <code>viewer.html</code> and <code>print.html</code> .
<code>images</code>	A directory containing the images referenced from <code>viewer.html</code> and <code>print.html</code> .

Distributed Mode

The report engine can be started as a daemon to which report applications can connect to process reports. One or more engines can be started on the same machine or on a different machine, hence *distributed mode*.

Distributed mode offers two advantages:

1. Vastly faster processing for short documents: The startup time of the JVM that is incurred for every report in regular mode can exceed the processing time of the report causing the overall performance to be poor. In distributed mode, the JVM is started and initialized only once. For report batches the improvement in performance is dramatic (A test has shown an increase by factor 24 so that a 500 file PDF batch completed in 17 seconds rather than in 7 minutes).
2. Improved scalability: Formatting graphical reports is CPU intensive. However CPU is usually expensive on the server optimized for IO that is running the DVM and/or the database. The distributed mode allows offloading the report processing entirely to another, very much cheaper, machine such as a

standard PC. Such a dedicated report formatting PC could be installed with Windows™ which has the additional advantage of handling fonts and printers in a user friendlier way.

To take advantage of distributed mode, you need to do the following:

1. Place your report design documents (.4rp) on the server where the GRE daemon runs.
2. Start the GRE daemon.
3. Provide the remote connection details in your source code.

Place the report design documents (.4rp) on the daemon server

Your report design documents need to be on the server running the daemon.

Start the Genero Report Engine daemon

The daemon is invoked by calling the script \$GREDIR/bin/greportwriter with the “-l” option and specifying the port it listens on. For example:

```
C:\Program Files\FourJs\Genero Studio\fgl>envcomp
C:\Program Files\FourJs\Genero Studio\fgl>cd ..\gre\bin
C:\Program Files\FourJs\Genero Studio\gre\bin>greportwriter -l 6500
```

If the Genero Report Engine daemon resides on a different machine than the DVM and you wish to preview reports, you must start the daemon with both the -l and -u options: C:\Program Files\FourJs\Genero Studio\gre\bin>greportwriter -l 6500 -u XXX

Connect to a daemon using the reporting API

The API function `fgl_report_configureDistributedProcessing` is used to select and configure distributed processing. It takes two parameters that denote the server to use; a host name and a port. This code shows an example for a GRE daemon running on the local machine (host “localhost”):

```
IF NOT fgl_report_loadCurrentSettings("OrderReport.4rp") THEN
  ...
END IF
...
CALL fgl_report_configureDistributedProcessing("localhost",6500)
RETURN fgl_report_commitCurrentSettings()
```

This example connects to a daemon running on the server “PrintServer”:

```
IF NOT fgl_report_loadCurrentSettings("OrderReport.4rp") THEN
  ...
END IF
...
CALL fgl_report_configureDistributedProcessing("PrintServer",6500)
RETURN fgl_report_commitCurrentSettings()
```

Logging in distributed mode

If the GREDEBUG environment variable is set, the daemon logs messages of the specified level to the file `gre.log` in the home directory of the user invoking the daemon.

The API function `fgl_report_setDistributedRequestingUserName` can be used to set a user name in the log file in order to distinguish between log entries originating from different users. See [fgl_report_setDistributedRequestingUserName](#) on page 631.

Previewing reports in distributed mode

Important: The following instructions are not necessary if the GRE and the DVM are running on the same physical machine.

To preview a report when the GRE daemon is running on a different machine than the DVM and the output type is either "PDF", "RTF", "XLS", "XLX" or "HTML" (as specified by the `fgl_report_selectDevice` function), you must do two things:

1. Start the Web server service by invoking the command `grehttpd`.

Located at `GREDIR/bin/grehttpd`, the following options are available:

- `-p port`, where `port` is the port number. The default port is 8080.
- `-q` activates the Web server service in quiet mode.
- `-d directory`, where `directory` specifies the Web root. There may be several. If nothing is specified, then the current working directory (".") is taken as the Web root.
- `-license` prints the NanoHTTPD license

Note: The Genero Report Engine ships with this minimal Web server, however any other web server capable of serving static files can be used.

2. Set the `GREOUTPUTDIR` environment variable to the specified Web root directory in the environment of the GRE daemon (`greportwriter -l`).

Genero Report Engine error messages

Table 232: Error messages for the Package `designtime.trigger`

Number	Description
GS-37400	Missing "name" attribute on "input-variable" element
GS-37401	Missing "type" attribute on "input-variable" element
GS-37402	Unrecognized "type" value "%1" on "input-variable" element
GS-37403	Missing "expectedLocation" attribute on "input-variable" element
GS-37404	Unrecognized "expectedLocation" value "%1" on "input-variable" element
GS-37405	Missing "name" attribute on "match" element
GS-37406	Missing "nameConstraint" attribute on "match" element

Table 233: Error messages for the Package `designtime`

Number	Description
GS-37200	Failed to find PXML node in template
GS-37206	Failed to find RTL "stylesheet" element in template
GS-37207	Failed to find element named "%1" in schema
GS-37208	Failed to find match node named "%1" in template
GS-37209	Failed to find root element "%1" in schema file "%2"
GS-37210	Failed to find RTL root match node in template

Table 234: Error messages for the Package layoutnode

Number	Description
GS-31200	Internal error: Invalid bar code type
GS-31201	<p>Bar code type "%1": invalid codeValue "%2". Value is not an integer</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31202	<p>Bar code type "%1": invalid codeValue "%2". The value needs to be %3 digits long</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31203	<p>Bar code type "%1": invalid checksum %2 in %3. Value should be %4</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31204	<p>DocumentStructureException: failed to add text node (cause=%1).</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31205	<p>DocumentStructureException: failed to finalize text node (cause=%1).</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31206	<p>Bar code type "%1": invalid codeValue "%2". Value is not an integer</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31207	<p>Bar code type "%1": invalid codeValue "%2". The value needs to be %3 digits long</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31208	<p>Bar code type "%1": invalid codeValue "%2". First digit must be either '0' or '1'</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31209	Internal error: Wrong initializer for digit "%1"

Number	Description
GS-31210	<p>Bar code type "Code128": invalid codeValue "%1". Value cannot be parsed as comma separated character list</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31211	<p>Bar code type "Code128": invalid start codeValue "%1". Sequence must start with STARTA, STARTB or STARTC</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31212	<p>Bar code type "%1": invalid checksum "%2" in %3. Value should be "%4"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31213	<p>Bar code type "Code128": invalid character "%1" in codeValue "%2". Code is not available in the character set %3.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31214	<p>Bar code type "%1": invalid codeValue "%2". Value is not an integer</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31215	<p>Bar code type "%1": invalid codeValue "%2". The value needs to be %3 digits long</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31216	<p>Bar code type "%1": invalid checksum %2 in %3. Value should be %4</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31217	<p>Bar code type "%1": invalid number of digits %2. Value must be a multiple of 2.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31218	<p>Bar code type "%1": invalid value for noCheckDigits %2. Value must be either 1 or 2.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Number	Description
GS-31219	<p>Bar code type "%1": invalid codeValue "%2". Value may contain only the digits 0-9 and the '-' character.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31220	<p>Bar code type "%1": invalid codeValue "%2". The value needs to be %3 digits long</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31221	<p>Bar code type "%1": invalid K checksum %2 in %3. Value should be '%4'</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31222	<p>Bar code type "%1": invalid C checksum %2 in %3. Value should be '%4'</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31224	<p>Bar code type "%1": invalid character '%2' in codeValue "%3".</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31225	<p>Bar code type "%1": invalid codeValue "%2". The value needs to be %3 digits long</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31226	<p>Bar code type "%1": invalid checksum %2 in %3. Value should be %4</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31227	<p>Bar code type "Code 39 Extended": invalid character "%1" in codeValue "%2".</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31228	<p>Bar code type "Code 39 Extended": invalid codeValue "%1". Value cannot be parsed as comma separated character list</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Number	Description
GS-31229	<p>Bar code type "Code 39 Extended": invalid character "%1" in codeValue "%2".</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31230	<p>Bar code type "%1": invalid codeValue "%2". Value is not an integer</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31231	<p>Bar code type "%1": invalid codeValue "%2". The value needs to be 9 digits long</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31232	<p>Bar code type "%1": invalid checksum %2 in %3. Value should be %4</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31233	<p>Internal error: Wrong initializer for digit "%1"</p>
GS-31234	<p>Bar code type "%1": invalid value for controlCharacters %2. Value must be a four character string.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31235	<p>Bar code type "%1": invalid value for noCheckDigits %2. Value must be either 1 or 2.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31236	<p>Bar code type "%1": invalid character '%2' in codeValue "%3".</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31237	<p>Bar code type "%1": invalid codeValue "%2". The value needs to be %3 digits long</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31238	<p>Bar code type "%1": invalid K checksum %2 in %3. Value should be '%4'</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Number	Description
GS-31239	<p>Bar code type "%1": invalid C checksum %2 in %3. Value should be '%4'</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31241	<p>Bar code type "Code 93 Extended": invalid codeValue "%1". Value cannot be parsed as comma separated character list</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31242	<p>Bar code type "Code 93 Extended": invalid character "%1" in codeValue "%2".</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31243	<p>Bar code type "%1": invalid character '%2' in codeValue "%3".</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31244	<p>Bar code type "%1": invalid codeValue "%2". The value needs to be %3 digits long</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31245	<p>Bar code type "%1": invalid codeValue "%2". The first character of the value needs to be either 'a', 'b', 'c' or 'd'</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31246	<p>Bar code type "%1": invalid codeValue "%2". The last character of the value needs to be either 't', 'n', '*' or 'e'</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31247	<p>Bar code type "%1": invalid codeValue "%2". The character '%3' may not be used within the value</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31248	<p>Bar code type "%1": invalid checksum %2 in %3. Value should be '%4'</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Number	Description
GS-31249	Internal error: Codabar 18: invalid check type
GS-31250	<p>Bar code type "%1": invalid codeValue "%2". The last character of the value needs to be either 'a', 'b', 'c' or 'd'</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31251	Internal error: Codabar 2: invalid check type
GS-31252	<p>Bar code type "%1": expected text to follow '\\' at end of "codeValue" attribute</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31253	<p>Bar code type "%1": expected three octal digits to follow '\\' in "codeValue" attribute at position %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31254	<p>Bar code type "%1": expected octal digit in "codeValue" attribute at position %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31257	<p>Bar code type "%1": code data too long (val=%2), max=928</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31258	<p>Bar code type "%1": invalid errorCorrectionDegree "%2". Value must be between 0 and 8.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31259	<p>Bar code type "%1": invalid value for lines "%2". Value must be a value between 1 and 90.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31260	<p>Bar code type "%1": invalid value for lines "%2". Value causes illegal value for dataSymbolsPerLine of %3 which must be a value between 1 and 30.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Number	Description
GS-31261	<p>Bar code type "%1": invalid dataSymbolsPerLine "%2". Value must be between 1 and 30.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31262	<p>Bar code type "%1": invalid value for dataSymbolsPerLine "%2". Value causes illegal value for lines of %3 which must be a value between 1 and 90.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31264	<p>Bar code type "%1": value for lines "%2" and dataSymbolsPerLine "%3" is too small to hold data of %4. The product must be at least %5</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31265	<p>Bar code type "%1": value for additionalPaddingLines "%2" causes the data lines to exceed the maximum value of 90 lines</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31266	<p>Bar code type "%1": encoding error: data needs to end with a latch to ASCII encoding</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31267	<p>Bar code type "data-matrix": invalid rawCodeValue "%1". Value cannot be parsed as comma separated character list</p>
GS-31268	<p>Character '%1' cannot be encoded since it is not available in the code page ISO-8859-1.</p>
GS-31269	<p>Datamatrix "Shifted ASCII" encoding: Character '%1' not allowed.</p>
GS-31270	<p>Datamatrix "Basic ASCII" encoding: Character '%1' not allowed.</p>
GS-31271	<p>Datamatrix "Shifted ASCII" encoding: Unknown encoding symbol "%1".</p>
GS-31272	<p>Datamatrix "Basic ASCII" encoding: Unknown encoding symbol "%1".</p>
GS-31273	<p>Datamatrix "C40 (basic set)" encoding: Character '%1' not allowed.</p>
GS-31274	<p>Datamatrix "C40 (basic set)" encoding: Illegal position of "STARTASCII" symbol.</p>
GS-31275	<p>Datamatrix "C40 (basic set)" encoding: Unknown encoding symbol "%1".</p>
GS-31276	<p>Datamatrix "C40 (set 1)" encoding: Unknown encoding symbol "%1".</p>
GS-31277	<p>Datamatrix "C40 (set 2)" encoding: Character '%1' not allowed.</p>
GS-31278	<p>Datamatrix "C40 (set 2)" encoding: Unknown encoding symbol "%1".</p>

Number	Description
GS-31279	Datamatrix "C40 (set 3)" encoding: Character '%1' not allowed.
GS-31280	Datamatrix "C40 (set 3)" encoding: Unknown encoding symbol "%1".
GS-31281	Datamatrix "C40" encoding: Encountered non data character as last character at the end of symbol space.
GS-31282	Datamatrix "C40" encoding: Encoded data exceeds maximum of 1558 bytes
GS-31284	Datamatrix "C40" encoding: Encountered non data character as last character at the end of symbol space.
GS-31285	Datamatrix "BASE 256" encoding: failed to parse "%1" as byte value.
GS-31286	Datamatrix "BASE 256" encoding: failed to parse "%1" as byte value (value is outside of the range 0-255).
GS-31287	Datamatrix "ANSI X11" encoding: Character '%1' not allowed.
GS-31288	Datamatrix "ANSI X11" encoding: Illegal position of "STARTASCII" symbol.
GS-31289	Datamatrix "ANSI X12" encoding: Unknown encoding symbol "%1".
GS-31290	Datamatrix "ANSI X12" encoding: Invalid number of encoded codewords (%1). The number must be a multiple of 3
GS-31291	Datamatrix "EDIFACT" encoding: Character '%1' not allowed.
GS-31292	Datamatrix "EDIFACT" encoding: Illegal position of "STARTASCII" symbol.
GS-31293	Datamatrix "EDIFACT" encoding: Unknown encoding symbol "%1".
GS-31294	Datamatrix "EDIFACT" encoding: Invalid number of encoded codewords (%1). The number must either be a multiple of 4 or terminated by a STARTASCII latch.
GS-31295	Code value too long
GS-31296	<p>Invalid number of arguments in gradient paint specification (Required are x1, y1, color1, x2, y2, color2, cyclic)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31297	<p>Failed to parse PXML x expression value %1 from the gradientPaint initializer %2. Parse Error:%3</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31298	<p>Failed to parse PXML y expression value %1 from the gradientPaint initializer %2. Parse Error:%3</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Number	Description
GS-31299	<p>Failed to parse PXML color value %1 from the gradientPaint initializer %2. Parse Error:%3</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31300	<p>Failed to parse boolean value "cyclic" from the gradientPaint initializer %1. Parse Error:%2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31301	<p>Invalid command "%1" in path "%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31302	<p>Unknown command "%1" in path "%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31303	<p>Number of items in xyList not a multiple of 2 (value=%1 count=%2 Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31304	<p>Must have same number of values as row titles multiplied by column titles . Have %1 values and %2 values Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31305	<p>Internal error: Invalid chart type</p>
GS-31306	<p>Syntax error int PXML font size expression (%1)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31307	<p>Failed to parse PXML fontsize expression "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31308	<p>Decimal format is not available for this input locale</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31309	<p>Decimal format is not available for this output locale</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31310	<p>Failed to create HTML view (cause=%1:%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31311	<p>Failed to load image from data (cause=%1)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31312	<p>Unsupported image format in data URL "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31313	<p>Unsupported image format loading image from URL "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31314	<p>Failed to load SVG document from file "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31315	<p>Failed to load image from file "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31316	<p>Unsupported image format loading image from file "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31317	<p>Failed to load image (cause="%1")</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31318	<p>Can't find resource "%1"</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31319	<p>Failed to load SVG document from resource URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31320	<p>Failed to load image from resource URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31321	<p>Failed to load "Base-64" encoded SVG document from data URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31322	<p>Failed to load SVG document from data URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31323	<p>Syntax error in data URL "%1" (Failed to locate delimiter ',')</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31324	<p>Unsupported encoding in data URL "%1" (Failed to find ";base64")</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31325	<p>Incomplete data URL "%1" (Data is missing)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31326	<p>Failed to load image from data URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31327	<p>Failed to load image from URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31329	<p>Invalid "anchorX" PXML expression value %1 computed from expression "%1" (must yield a value between 0 and 1))</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31330	<p>Invalid "anchorY" PXML expression value %1 computed from expression "%1" (must yield a value between 0 and 1))</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31331	<p>Invalid font style value "%1" in list "%2" (must be one of: "plain", "italic" or "bold")</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31332	<p>Failed to parse font size percent expression %1 (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31333	<p>The empty string is not allowed as PXML color expression</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31334	<p>Failed to parse PXML color expression "%1" (must be a '#' followed by an integer in hex notation) (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31335	<p>Invalid port "%1" value</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31336	<p>invalid value for lowWaterMark (must be integer >= 0) val=%1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31337	<p>invalid value for highWaterMark (must be negative or greater than lowWaterMark) lowWaterMark=%1 highWaterMark=%2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31338	<p>SAXException: failed to initialize (cause=%1).</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31339	Timeout: Page was not consumed within %1 milliseconds
GS-31340	<p>SAXException: failed to create page (cause=%1).</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31341	<p>SAXException: failed to finish document (cause=%1).</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31342	Internal error: Page not in map
GS-31343	<p>IOException: failed to transform memory queue into disk queue (cause=%1).</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31344	<p>IOException: failed to write page to disk queue (cause=%1).</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31345	<p>IOException: failed to flush queue to disk (cause=%1).</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31346	Internal error: %1::eventInitializedChild() called for a cloned LayoutNode
GS-31347	<p>Nodes attached to named ports may not be self placing</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31348	<p>All nodes attached to named ports must be specified prior to specifying any nodes for the primary port</p> <p>A <i>port</i> is the same as a <i>section</i>.</p> <p>A node is said to be attached to a named port in its parent if it has the Section property set.</p> <p>A node is said to be attached to the primary port if it does not have the Section property set.</p> <p>Node lists are ordered. If you look at the Report Structure view where nodes of the same parent are stacked vertically, a node is prior to another node when it is higher up.</p> <p>To solve this issue, reorganize the nodes such that all named nodes sit higher in the parent node than nodes attached to the primary port.</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31349	Internal error: cannot propagate, parent==null parent not instanceof LayoutNode
GS-31350	Internal error: have uninitializedChild when cloning
GS-31351	Internal error: %1::attachPortedChild() invalid port. Port=%2
GS-31352	<p>The "roundingMode" attribute is not supported on this platform (requires Java version 1.6 or higher).</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31353	<p>Failed to parse value "%1" using the specified input format (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31354	<p>Failed to format field value "%1" using format "%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31355	<p>Invalid page number format "%1" (must be one of : "arabic", "lowerroman", "roman" or "upperroman")</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31356	Illegal argument value "%1" as second parameter in call to the PXML function format(). Valid values are ARABIC, LOWERROMAN and UPPERROMAN
GS-31357	Internal error: getStringReturnValue called on function format. The function returns a string value
GS-31358	Internal error: getParameterType called on function %1. The function takes no parameters
GS-31359	Internal error: pushString called on function %1. The function takes no parameters
GS-31360	Internal error: pushNumeric called on function %1. The function takes no parameters
GS-31361	Internal error: getStringReturnValue called on function %1. The function returns a numeric value
GS-31362	Internal error: pushString called on function %1. The function takes only one string parameter
GS-31363	Internal error: getStringReturnValue called on function %1. The function returns a numeric value
GS-31364	Number of sizes and titles differ (Have %1 sizes and %2 titles)

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31365	Internal error: Invalid pie graph type
GS-31366	Internal error: %1::eventNewChild() called for a cloned LayoutNode
GS-31367	Internal error: %1::eventInitializedChild() called for a cloned LayoutNode
GS-31368	Internal error: cannot propagate, parent==null parent not instanceof LayoutNode
GS-31369	Internal error: have uninitializedChild when cloning
GS-31370	Loading SVG document from url %1 failed (cause=%2)
GS-31371	Loading SVG document from reader failed (cause=%1)
GS-31372	Loading SVG document from input stream failed (cause=%1)
GS-31373	<p>Invalid command in transformInstructions attribute ("%1")</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31374	<p>not enough arguments to %1 command in transform attribute. Required are %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31375	<p>Wrong number of matrix values in %1 attribute (%2) required are 6</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31376	Internal error: %1 expression=%2
GS-31377	<p>Indentation value exceeds width by %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31378	Internal error: No parent
GS-31379	Internal error: Parent not of type LayoutNode (type=%1)
GS-31382	<p>The number of items in all series must be the same for the "StackedArea" type.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31383	<p>The x values of different series need to be the same for the "StackedArea" type.</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31384	<p>The number of xValues and yValues differ (Have %1 xValues and %2 yValues)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31385	<p>When seriesTitles are specified, they must have same number as xValues and yValues. Have %1 xValues and %2 seriesTitles</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31386	Internal error: Invalid code type for XYGraph
GS-31388	Internal error: UnsupportedEncodingException "%1" for value "%2" using encoding "%3"
GS-31389	<p>QR-Code: encoding exception "%1" on encoding value "%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31391	<p>QR-Code: Unsupported encoding "%1" (Run "CharsetInfo" for a list of supported encodings)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31392	<p>QR-Code: Invalid the error correction degree value %1 (must be a value between 0 and 3)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31393	<p>Invalid font size value %1 (value needs to be to be greater than 0)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31394	<p>Error parsing hex digit {0} at position %2 in string "(2)"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31395	Parse error: Failed to parse tracking code and routing code from code value "%1" (Found more than one comma)

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31396	<p>Parse error: Failed to parse tracking code and routing code from code value "%1" (Encountered non digit character {1})</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31397	<p>Parse error: routing code "%1" exceeds 11 digits</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31398	<p>Parse error: Tracking code "%1" is too short (needs to have at least 2 digits)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-31399	<p>Parse error: The second digit of the tracking code {0} is outside the allowed range 0 - 4 in tracking code "%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Table 235: Error messages for the Package main

Number	Description
GS-32100	<p>Instantiating the Viewer failed with a TargetInvocationException (cause=%1:%2)</p>

Table 236: Error messages for the Package shared

Number	Description
GS-32700	<p>attribute %1 must be specified</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32701	<p>Invalid value "%1" (must be integer)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32702	<p>Invalid value "%1" (must be a floating point value)</p>

Number	Description
	<p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32703	<p>Invalid value "%1" (must be boolean)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32704	<p>Invalid enumeration value "%1" (must be one of: %2)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32705	<p>Invalid option value "%1" (must be one of: %2)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32706	<p>Required option value (must be one of: %1)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32707	<p>Invalid mandatory option value "%1" (must be one of: %2)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32708	<p>Invalid boolean option value "%1" (must be one of: %2)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32709	<p>Required boolean option value (must be one of: %1)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32710	<p>Invalid mandatory boolean option value "%1" (must be one of: %2)</p>

Number	Description
	<p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32711	<p>Invalid value "%1" (must be a colon separated list of float point values)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32712	<p>Invalid value "%1" (must be a colon separated list of POSITIVE float point values)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32713	<p>Invalid value "%1" (must be a colon separated list of integer values)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32714	<p>Invalid object option value "%1" (must be one of: %2)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32715	<p>Required object option value (must be one of: %1)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32716	<p>Invalid mandatory object option value "%1" (must be one of: %2)</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32717	<p>Invalid value "%1" (must be a colon separated list of strings): unexpected character "%2" at position %3. Expected a " " or "," character</p> <p>Error message and resolution should be self-explanatory.</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-32718	<p>Invalid value "%1" (must be a colon separated list of strings): encountered unclosed string starting a position %2.</p> <p>Error message and resolution should be self-explanatory.</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Table 237: Error messages for the Package stylesheet.rtl

Number	Description
GS-33900	<p>Unknown RTL variable type "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33901	<p>Failed to initialize numeric RTL attribute variable from the input string "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33902	<p>Missing "url" attribute in RTL call-report element at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33903	<p>Missing RTL match element at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33904	<p>Missing RTL "name" attribute in entity declaration at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33905	<p>Missing "url" attribute in RTL load-entities element at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33906	<p>Missing "name" attribute in RTL expand-entity element at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Number	Description
GS-33907	<p>Reference to undeclared RTL entity "%1" at %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33908	<p>Failed to replace RTL entity parameter "%1". Parameter was not specified at %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33909	<p>Missing "name" attribute in RTL entity parameter element at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33910	<p>Missing "value" attribute in RTL entity parameter element at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33911	<p>Failed to find RTL style sheet root element at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33912	<p>Unknown RTL element %1 at %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33913	<p>Encountered style sheet element with more than one RTL element at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33914	<p>Missing RTL elements in a sequences block at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33915	<p>Missing RTL alternatives in a or block at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33916	<p>Value for attribute %1 cannot be parsed as integer (Value=%2) at RTL element %3</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33917	<p>Invalid RTL attribute value selector %1 in attribute constraint %2 at %3</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33918	<p>Failed to resolve namespace prefix %1 in RTL constraint for %2 at %3</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33919	<p>Failed to parse regular expresssion for RTL attribute %1. Error=%2 at %3</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33920	<p>RTL attribute "name" not specified at "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33921	<p>RTL attribute "type" not specified at "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33922	<p>Illegal RTL location value "%1" at %2 (must be one of "expectedHere", "expectedBefore", "expectedAhead" or "expectedWayAhead")</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33923	<p>Error in input to RTL style: got token opening tag %1 when expecting %2</p>
GS-33924	<p>Error in input to RTL style: got closing tag %1 when expecting %2</p>
GS-33925	<p>Expression error: %1 evaluating RTL expression "%2" at attribute "%3" at %4</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33926	<p>Expression error: No variable value for variable "%1" encountered in input stream when evaluating RTL expression "%2" at attribute "%3" at %4</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33927	<p>Expression error: %1 evaluating RTL expression "%2" at attribute "%3" at %4</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33928	<p>Type error: Expression %1 yielded null instead of a %2 value at RTL attribute %3 at %4</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33929	<p>Type error: Expression %1 yielded %2 instead of a %3 value at RTL attribute %4 at %5</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33930	Reference to unknown global RTL method "%1"
GS-33931	Unknown RTL method %1 in the current context of type %2
GS-33932	Unknown RTL method %1 on object of type %2
GS-33933	Wrong number of arguments supplied to method "%1.substring" (Expected 2 or 3, got %2)
GS-33934	%1 to %2 conversion error in RTL function "%3.toNum()" (cannot parse value "%4" as %5)
GS-33935	Illegal access error. Not able to call RTL method "%1" on object of type %2 (cause=%3)
GS-33936	Error executing RTL method "%1" on object of type %2 (cause=%3)
GS-33937	Wrong number of args supplied to RTL method "%1.%2" (Expected %3, got %4)
GS-33938	Invalid type for argument no %1 in call to RTL method "%2.%3" (Expected %3, got %4)
GS-33939	Invalid null argument for argument no %1 in call to RTL method "%2"
GS-33940	Illegal access error. Not able to obtain value of RTL variable %1 on object of type %2 (cause=%3)
GS-33941	Internal error: Attempt to reference an RTL array element on a null pointer (key="%1")
GS-33942	Attempt to reference an RTL array element on a non array object of type %1 (key="%2")
GS-33943	Internal RTL error: Can't find character context
GS-33944	RTL attribute "name" not specified in rtl:array element

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33945	<p>RTL attribute "key" not specified in rtl:array element</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33946	<p>RTL attribute "name" not specified in rtl:let element at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33947	<p>RTL attribute "value" not specified in rtl:let element at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33948	<p>Undeclared RTL variable %1 at %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33949	<p>RTL attribute "name" not specified at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33950	<p>Expression error "%1" retrieving value of RTL variable "%2" at %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33951	<p>Unknown RTL variable %1 at %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33952	<p>RTO variable %1 in rtl:array-let element is not of type array but of type %2 at %3</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33953	<p>RTL attribute "key" not specified at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33954	<p>RTL expression "%1" in attribute "key" evaluated to null at %2</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33955	<p>RTL attribute "name" not specified at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33956	<p>RTL expression "%1" in attribute "key" evaluated to null at %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33957	<p>RTL attribute "name" not specified at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33958	<p>Internal RTL error: Failed to find "toCharacter"</p>
GS-33959	<p>Encountered RTL "%1" element containing an invalid element "%2" at %3</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33960	<p>missing "enumeration" attribute value in RTL "for-each" element encountered at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33961	<p>Invalid object type %1 returned for RTL attribute "enumeration" in a "for-each" element. Expected expression "%2" to return enumeration type at %3</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33962	<p>Missing RTL "condition" attribute value in "while" element encountered at %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33963	<p>Internal RTL error: Order of calls violated</p>
GS-33964	<p>Illegal trailing '.' in RTL identifier</p>
GS-33965	<p>Redefinition of RTL identifier %1 of type %2 as record</p>
GS-33966	<p>%1 to %2 conversion error in RTL function "%3.toNumeric()" (cannot parse value "%4" as %5)</p>

Number	Description
GS-33967	Date.fromString(): Failed to parse date value "%1" with pattern "%2"
GS-33968	Date.parseString(): Failed to parse date value "%1" with pattern "%2"

Table 238: Error messages for the Package stylesheet.standardpipe

Number	Description
GS-33800	Cannot open file Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33802	Old file format. Use last version of Genero Report Designer to update the file Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33803	File format : " + version + " not supported. This executable supports : " + SUPPORTED_VERSION_T Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33804	logical page mapping is set to "labels" but fgl_report_configureLabelOutput() was not called to configure the layout Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33805	cannot determine label width. Value was not set in call to fgl_report_configureLabelOutput() and it is not set in the .4rp template. Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33806	cannot determine label height. Value was not set in call to fgl_report_configureLabelOutput() and it is not set in the .4rp template. Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33807	environment variable GREDIR is not set Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33808	environment variable FGLDIR is not set

Number	Description
	Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.

Table 239: Error messages for the Package stylesheet

Number	Description
GS-33000	%1: not prepared to handle nested style declarations
GS-33001	Failed to parse style attribute "%1". Parse error=%2 Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33002	Error parsing CSS stylesheet: %1
GS-33003	Internal error: Node %1 (%2) getIdValue() failed
GS-33004	Internal error: Invalid operator %1
GS-33005	Internal error: Invalid Id %1
GS-33006	Internal error: Invalid Id %1
GS-33007	Internal error: failed to find simple selector index
GS-33008	Internal error: garbled selector
GS-33009	error during printing: Exception: %1
GS-33010	Executing process "%1" returned non zero exit code: %2, Process output:%3 Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33011	Got an exception of type "%1" when attempting to start process "%2" (cause=%3) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33012	IOException writing to file "%1" (cause="%2") Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33013	Internal error: closing tag="%1"
GS-33014	Internal error: Expected closing '%1' tag. Got "%2" tag instead
GS-33015	Please contact your sales office regarding licensing.
GS-33016	Missing COLDEF section in TABLE
GS-33017	Found illegal tag "%1" within COLDEFS section
GS-33018	COLDEF element requires "fWidth" or "pWidth" attribute

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33019	COL references undeclared column in COLDEFS section
GS-33020	<p>Illegal align value "%1" (must be one of "left", "right", "center" or "baseline")</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33021	<p>invalid value %1 for align attribute (must be one of "left", "right" or "center")</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33022	LI element is not nested within a OL or UL element
GS-33023	Encountered DIMENSIONS element at unexpected location within a PIVOTTABLE. DIMENSIONS need to be declared before any VALUES.
GS-33024	Encountered a DATACOLUMN element for an undeclared column.
GS-33025	%1 element encountered at unexpected position.%2
GS-33026	<p>Failed to parse value "%1" as floating point value</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33027	<p>Not enough dimensions in pivot table to draw as a map chart (Table needs to have at least one dimension column)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33028	<p>Not enough values in pivot table to draw as a map chart (Table needs to have at least one value column)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33029	<p>Not enough dimensions in pivot table to draw as a map chart (Table needs to have at least two dimension columns)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33030	Not enough values in pivot table to draw as a map chart (Table needs to have at least one value column)

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33031	<p>Not enough values in pivot table to draw as a xy chart (Table needs to have at least two value columns)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33032	<p>First value is not numeric (The first two values need to be numeric in order to draw a XY chart)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33033	<p>Second value is not numeric (The first two values need to be numeric in order to draw a XY chart)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33034	Internal error: closing tag="%1"
GS-33035	Internal error: Expected closing "%1" tag. Got "%2" tag instead
GS-33036	Internal error: Reference to unregistered color "%1"
GS-33037	Internal error: Reference to unregistered font "%1"
GS-33038	Incomplete document
GS-33039	<p>IllegalAccessException: Cannot load class "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33040	<p>InstantiationException: Cannot load class "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33041	<p>NoSuchMethodException: Cannot load class "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33042	<p>InvocationTargetException: Cannot load class "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33043	Internal error: IOException (cause="%1"

Number	Description
GS-33044	Internal error: parse called
GS-33045	Syntax error in PXML font size expression (%1) (Missing comma) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33046	Syntax error in PXML font size expression (%1) (can't parse "line" and "column" values as integers) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33047	Failed to parse font size percent expression %1 (cause=%2) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33048	Internal error: Could not find element %1 with source id %2 (name=%3), path=%4
GS-33049	Internal error: Could not find element %1 with source id %2 (name=%3), path=%4
GS-33050	Internal error: Failed to retrieve parent font size: (cause=%1)
GS-33051	Invalid port %1 Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33052	Invalid page number format "%1" Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33053	Internal error: Got exception of type "%1" during creation of an hex encoded bitmap (cause=%2) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33054	COLDEF element requires "fWidth" or "pWidth" attribute Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-33055	Internal error: failed to find child %1 for parent %2
GS-33056	Internal error: node is of unexpected type %1
GS-33057	COL references undeclared column in COLDEFS section in row %1
GS-33058	Exception of type "%1" encountered when trying to load image from URL "%2" (cause=%3)

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33059	<p>Value may not contain the file name separator character '%1'</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33060	<p>Value "%1" is not supported on this platform (Run "java ImageIOInfo" to obtain a list of available formats)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33061	<p>Got an exception of type "%1" when attempting to start the PPM image processor "%2" (cause=%3)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33062	<p>Got an exception of type "%1" when attempting to write to the PPM image processor "%2" (cause=%3)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33063	<p>Executing PPM processor "%1" returned non zero exit code: %2 (Process output=%3)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33064	<p>Executing PPM processor "%1" returned non zero exit code: %2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33065	<p>Failed to save image to file "%1" (cause=%2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33066	<p>Encountered exception of type "%1" while attempting to save a page image to a stream (cause=%2}</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33067	<p>Failed to save PPM image to file "%1" (cause=%2}</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33068	<p>Invalid value %1 (Allowed values are between 0 and 1)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33069	<p>Failed to save PDF document to file "%1" (cause=%2}</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33070	<p>Encountered exception of type "%1" when attempting to write PDF document to host "%2" at port %3 (cause=%4)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33071	<p>Encountered '{' in list (no nested lists allowed) at character position %1 in data "%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33072	<p>Unexpected character '%1' at character position %2 in data "%3"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33076	<p>Encountered unexpected '{' character at character position %1 in data "%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33078	<p>Missing '{' to close list at character position %1 in data "%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33079	<p>Encountered unfinished attribute declaration at character position %1 in data "%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33080	<p>Missing '=' character after attribute name at character position %1 in data "%2"</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33081	<p>Missing attribute value after '=' character at character position %1 in data "%2".</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33082	<p>%1: failed to write to host "%2" at port %3, reason: %4</p>
GS-33083	<p>Unknown encoding %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33084	<p>Invalid value '%1' for attribute %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33085	<p>Internal error: got exception of type "%1" when creating a "date time" attribute from value "%2" (cause=%3)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33086	<p>Internal error: got exception of type "%1" when creating an integer attribute from value "%2" (cause=%3)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33087	<p>Internal error: got exception of type "%1" when creating a text attribute from value "%2" (cause=%3)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33088	<p>Attribute "mediaTrayNumbers" is only available on the Linux platform</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33090	<p>No print service found</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33094	<p>Cannot write to file %1 msg=%2</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33095	Internal error: Failed parse pattern value "%1". Msg=%2
GS-33096	<p>Got an exception of type "%1" when attempting to launch the spooler command "%2" (cause=%3)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33097	<p>Got an exception of type "%1" when attempting to write to the spooler "%2" (cause=%3)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33098	<p>cannot find stream printer for mime type "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33100	Internal error: reject called
GS-33101	Internal error: got non 2D Graphics
GS-33102	Internal error: pageIndex %1 out of range (%2,%3)
GS-33103	Unexpected element "%1". Expected document root element to be a "PXML" element.
GS-33104	%1 root element has wrong namespace "%2". Expected namespace is "%3". Try adding xmlns="%4" to the root element.
GS-33105	<p>Class %1 not found</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33106	<p>Cannot load class %1 (%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33108	<p>Cannot load class %1. constructor not found (%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33110	Internal error: %1::endElementEx() current==null
GS-33117	IOException on writing RTF document to file "%1" (cause=%2)

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33118	%1 element is not nested within a %2 element
GS-33120	setting up print service failed (%1)
GS-33121	IO error during printing (%1), cause=%2
GS-33122	aborted printing (%1)
GS-33123	error during printing (%1:%2)
GS-33124	error during printing, printing aborted.
GS-33125	error during printing (%1)
GS-33126	internal error: got invalid value %1 for property orientationRequested. Setting to %2.
GS-33127	<p>InvocationTargetException: failed to instantiate style sheet "%1" (cause=%2).</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33128	<p>Exception: failed to instantiate style sheet "%1" (cause=%2).</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33129	IOException reading compressed source (cause=%1)
GS-33131	<p>Invalid command in transform attribute ("%1")</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33132	<p>Not enough arguments to %1 command in transform attribute. Required are %2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33133	<p>Cannot parse argument "%1" as number for the %2 command in transform attribute</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33134	<p>Wrong number of matrix values in %1 attribute (%2) required are %3</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Number	Description
GS-33135	<p>Uneven number of values in %1 attribute (%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33136	<p>Cannot resolve reference in use tag href=%1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33137	<p>Internal error: Invalid end cap value %1</p>
GS-33138	<p>Internal error: Invalid line join %1</p>
GS-33139	<p>Attribute value error: %1 must be a value between %2 and %3 (val=%4)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33140	<p>Wrt error: failed to read from source</p>
GS-33141	<p>Wrt error: Got non zero status value:%1, msg=%2</p>
GS-33142	<p>Wrt error: Exception=%1, msg="%2"</p>
GS-33143	<p>IOException writing to file "%1". msg="%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33144	<p>IOException flushing to temp file. msg="%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33145	<p>The TransformerFactory does not support SAX input and SAX output</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33146	<p>Encountered "TransformerConfigurationException" trying to initialize for loading XSLT style from URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33147	<p>Encountered "SAXException" trying to initialize for loading XSLT style from URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33148	<p>Encountered "IOException" trying to load XSLT style from URL "%1" (cause=%2)</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33150	<p>Encountered "TransformerConfigurationException" trying to load XSLT style from URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33151	<p>Encountered "SAXNotRecognizedException" trying to load XSLT style from URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33152	<p>Encountered "SAXNotSupportedException" trying to load XSLT style from URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33153	<p>Encountered "SAXException" on issuing "startDocument" using XSLT style from URL "%1" (cause=%2)</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33154	<p>Missing column value for column %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33155	<p>Illegal character set name %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33156	<p>Unsupported character set name %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33157	<p>Failed to write page number template (cause=%1}</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33158	<p>Path %1 is not a directory</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33159	<p>Failed to create directory %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33160	<p>Unexpected element "%1". Expected document root element to be a "SVG" element.</p>
GS-33161	<p>Encountered unreferenceable "g" element in "defs" element ("id" attribute missing).</p>
GS-33162	<p>Encountered unreferenceable "font" element in "defs" element ("id" attribute missing).</p>
GS-33163	<p>Encountered unreferenceable "image" element in "defs" element ("id" attribute missing).</p>
GS-33164	<p>Encountered faulty "image" element in "defs" element ("href" attribute missing).</p>
GS-33165	<p>Failed to create image file in directory %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33166	<p>Failed to create file "index.txt" in directory %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33167	<p>font id "%1" does not end with a style digit.</p>
GS-33168	<p>Failed to create WOFF font file "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33169	<p>Failed to create TTF font file "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33170	<p>Failed to create EOT font file "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33171	<p>Failed to rename file %1.part</p>

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33172	<p>Failed to create generic report design document from template %1</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33173	<p>Failed to create temporary generic report design document file</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33174	<p>Printer %1 not found</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33175	<p>Failed to rename WOFF font file to "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33176	<p>Failed to rename TTF font file to "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33177	<p>Failed to close image file "%1" (cause=%2}</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33178	<p>IOException closing file "%1". msg="%2"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33179	<p>Failed to rename EOR font file to "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-33180	<p>Failed to create file "%1"</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>

Table 240: Error messages for the Package util.domutil.selection

Number	Description
GS-34800	Internal error: Can't find element denoted by the path %1 in the document fragment %2

Table 241: Error messages for the Package util.domutil

Number	Description
GS-34500	Internal error: Element.getLocalName() returned null. Apparantly the document was modified using DOM Level 1 functions
GS-34501	Attribute type error: Expected integer value for attribute %1 Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34502	Attribute type error: Expected long value for attribute %1 Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34503	Attribute type error: Expected double value for attribute %1 Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34504	Attribute type error: Expected boolean value for attribute %1 Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34505	Failed to parse value "%1" as URL (cause=%2) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34506	Cannot resolve relative URL %1. Have no system id to absolutize against. Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34507	Failed to parse "systemId" value "%1" as URI (cause=%2) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34508	Failed to create absolute URL from systemId="%1" and URL="%2" Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.

Number	Description
GS-34509	Internal error: failed to convert URI "%1" to URL (cause=%2) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34510	Internal error: location path contains unparseable value (path=%1).
GS-34511	Cannot load document %1. Have "FactoryConfigurationError" (cause=%2) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34512	Cannot load document %1. Have "ParserConfigurationError" (cause=%2) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34513	Cannot load document %1. Have "SAXParseException" (cause=%2) at line %3 Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34514	Cannot load document %1. Have "SAXException" (type=%2, cause=%3) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34515	Cannot load document %1. Have "IOException" (cause=%2) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-34516	Encountered "IOException" trying obtain input stream for URL "%1" (cause=%2) Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.

Table 242: Error messages for the Package util.expressionparser

Number	Description
GS-35400	Internal error: pushString called on function %1. The function takes only one numeric parameter
GS-35401	Internal error: getStringReturnValue called on function %1. The function returns a numeric value
GS-35402	Internal error: pushString called on function %1. The function takes only two numeric parameters

Number	Description
GS-35403	Internal error: getStringReturnValue called on function %1. The function returns a numeric value
GS-35404	RTL expression error: Illegal access error. Not able to obtain value of variable %1 on object of type %2 (cause=%3)
GS-35405	RTL expression error: Illegal array or array index. Not able to obtain array value for index %1 on List of type %2 (err=%3)
GS-35406	RTL expression error: Illegal array index. Not able to obtain array value for index %1 on List of type %2 (cause=%3)
GS-35407	RTL expression error: Unable to resolve non numeric key %1 on array of type %2 (Non numeric arguments can only be used on "Map" arrays)
GS-35408	Unknown media type "%1" Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-35409	Unknown media type "%1" Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-35410	Unexpected extra text %1 at end of PXML expression "%2"
GS-35411	Unexpected extra text %1 at end of PXML string expression "%2"
GS-35412	Unexpected token %1 in PXML expression. Expected else operator ':' in ternary conditional expression
GS-35413	Unexpected token %1 following ' ' in PXML expression
GS-35414	Unexpected token %1 following '&' in PXML expression
GS-35415	Unexpected token %1 following '%2' in PXML expression
GS-35416	Unexpected end of PXML expression
GS-35417	Type error: illegal call to PXML function "%1" returning %2 while expecting a function that returns a numeric value
GS-35418	Expected enumeration identifier for parameter number %1 in call to PXML function %2 but got %3 instead
GS-35419	Missing function arguments in call to PXML function %1. Expected %2 parameters but found %3 instead
GS-35420	Missing ')' in call to PXML function "%1"
GS-35421	Unmatched '(' in PXML expression
GS-35422	Internal error in PXML expression (unexpected token %1)
GS-35423	Unexpected end of PXML string expression
GS-35424	Type error: illegal call to PXML function "%1" returning %2 when expecting a function that returns a string value

Number	Description
GS-35425	Expected enumeration identifier for parameter number %1 in call to PXML string function "%2" but got %3 instead
GS-35426	Missing function arguments in call to PXML string function "%1" (Expected %2 parameters but found %3)
GS-35427	Missing ')' in call to PXML string function "%1" Error message and resolution should be self-explanatory.
GS-35428	Unmatched '(' in PXML string expression Error message and resolution should be self-explanatory.
GS-35429	Invalid operator in expression: '%1' (%2)
GS-35430	Unexpected extra token "%1" at end of RTL expression "%2"
GS-35431	Unexpected token %1 in RTL expression. Expected else operator ':' in ternary conditional expression
GS-35432	Error in RTL conditional expression: Expected %1 value to precede '?' operator but got a %2 value
GS-35433	Error in RTL conditional or expression: Expected left operand to be of type %1 but got a %2 value
GS-35434	Error in RTL conditional or expression: Expected right operand to be of type %1 but got a %2 value
GS-35435	Error in RTL conditional and expression: Expected left operand to be of type %1 but got a %2 value
GS-35436	Error in RTL conditional and expression: Expected right operand to be of type %1 but got a %2 value
GS-35437	RTL expression error: Relational operator ">=" cannot be applied to types %1,%2 Error message and resolution should be self-explanatory.
GS-35438	RTL expression error: Relational operator ">" cannot be applied to types %1,%2 Error message and resolution should be self-explanatory.
GS-35439	RTL expression error: Relational operator "<=" cannot be applied to types %1,%2 Error message and resolution should be self-explanatory.
GS-35440	RTL expression error: Relational operator "<" cannot be applied to types %1,%2 Error message and resolution should be self-explanatory.
GS-35441	RTL expression error: Operator '+' cannot be applied to %1,%2 Error message and resolution should be self-explanatory.
GS-35442	RTL expression error: Operator '-' cannot be applied to %1,%2

Number	Description
	Error message and resolution should be self-explanatory.
GS-35443	RTL expression error: Operator '*' cannot be applied to %1,%2 Error message and resolution should be self-explanatory.
GS-35444	RTL expression, type mismatch: cannot multiply value of type %1 Error message and resolution should be self-explanatory.
GS-35445	RTL expression, division failed (cause=%1) Error message and resolution should be self-explanatory.
GS-35446	RTL expression: Operator '/' cannot be applied to %1,%2 Error message and resolution should be self-explanatory.
GS-35447	RTL expression, integer division failed (cause=%1) Error message and resolution should be self-explanatory.
GS-35448	RTL expression error: Operator '%' cannot be applied to %1,%2 Error message and resolution should be self-explanatory.
GS-35449	RTL expression error: Illegal array index value (null)
GS-35450	RTL expression error: Unexpected token %1 ("%2" following array index expression. Expected ']' to terminate expression
GS-35451	RTL expression error: Cannot negate object of type %1
GS-35452	RTL expression error: Cannot perform logical negation on an object of type %1
GS-35453	Unexpected end of RTL expression
GS-35454	RTL expression error: Unexpected token %1 ("%2" in call to function %3. Expected either ',' or ')').
GS-35455	RTL expression error: Unmatched '('
GS-35456	RTL expression error: Unexpected token %1 (type=%2)
GS-35457	ExpressionTokenizer: Internal error: pushBack() called more than once in a row.
GS-35458	Invalid character '%1' in expression "%2"
GS-35459	Unexpected end of expression in expression "%1", was expecting '%2'
GS-35460	Unexpected character '%1' in expression "%2", was expecting '%3'
GS-35461	Failed to parse numeric value "%1"
GS-35462	Failed to parse "%1" as a numeric value for the RTL variable "%2" Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-35463	Unknown function "%1"

Number	Description
GS-35464	Unknown function "%1()"
GS-35465	Unknown function "%1(%2)"
GS-35466	Unknown PXML unit "%1"
GS-35467	Unknown variable "%1"
GS-35468	Unexpected token "" + TOKEN_NAMES[ttype] + "" in PXML string expression

Table 243: Error messages for the Package util.getopt

Number	Description
GS-36000	Invalid '=' character in option "%1"
GS-36001	Internal error: Invalid usage
GS-36002	Invalid option "%1". Expected '-' to start option
GS-36003	Invalid option "%1". Expected option name to follow '-'
GS-36004	Invalid option "%1". Expected option name to follow "--"
GS-36005	Unknown option "%1"
GS-36006	Missing required argument for option "%1".

Table 244: Error messages for the Package util.regexp.derivates

Number	Description
GS-36600	Unknown external reference %1
GS-36601	Internal error: Tokenizer returned invalid values n=%1
GS-36603	Internal error: Tokenizer returned invalid values m=%1
GS-36604	Internal error: Tokenizer returned invalid values n=%1 m=%2
GS-36606	Internal error: got internal reference token although the stream claims not to produce these
GS-36607	Reference index %1 out of range (max=%2
GS-36608	found recursive external reference %1. Stack=%2
GS-36609	Line does not consist of two space separated fields Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-36610	Expression file is empty Tip: This is an <i>attribute error</i> . Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.
GS-36611	Unexpected end of expression

Number	Description
GS-36613	Unexpected EOF in control character specification
GS-36614	Illegal control character specification "\c%1"
GS-36615	Unexpected EOF in external reference
GS-36616	Expected '{' character to follow \R in external reference. Got '%1' instead
GS-36617	Unclosed external reference
GS-36618	Unexpected EOF in tracked external reference
GS-36619	Expected '{' character to follow \R in a tracked external reference. Got '%1' instead
GS-36620	Unclosed tracked external reference
GS-36621	Unexpected EOF in marker definition
GS-36622	Expected '{' character to follow \M in marker definition. Got '%1' instead
GS-36623	Unclosed marker definition
GS-36627	Expected '{' character to follow \p in character class definition. Got '%1' instead
GS-36628	Unclosed character class definition
GS-36629	Unknown character class name %1
GS-36630	Invalid escape "\%1"
GS-36631	Unclosed character class
GS-36632	illegal digit '%1' in range modifier
GS-36633	Unclosed range modifier
GS-36634	Upper range value smaller than lower range value in range modifier
GS-36635	Invalid range values (%1,%2) in range modifier
GS-36636	Invalid range value (%1,) in range modifier
GS-36637	Invalid range value (,%1) in range modifier
GS-36638	Illegal range modifier
GS-36639	Empty range modifier
GS-36640	Failed to find an element definition
GS-36641	Failed to find referenced element
GS-36642	Failed to find complexType element
GS-36643	Failed to find simpleType element
GS-36644	Failed to find attribute declaration
GS-36645	Failed to find attribute group declaration
GS-36646	Failed to find group declaration
GS-36647	Neither "ref" nor "name" Attributes not set on attribute group

Number	Description
GS-36648	Neither "ref" nor "name" Attributes not set on attribute
GS-36649	Encountered empty simpleType
GS-36650	Encountered extension without base declaration
GS-36651	Encountered restriction without base declaration
GS-36652	Neither "ref" nor "name" Attributes not set on element
GS-36653	maxOccurs has value 0 in particle
GS-36654	maxOccurs is smaller than minOccurs in particle
GS-36656	Internal error: Unknown type %1

Table 245: Error messages for the Package util.regex

Number	Description
GS-36300	Unexpected extra text at end of expression at %1 (c='%2')
GS-36301	Unexpected end of expression
GS-36302	Missing ']' at %1
GS-36303	Missing ')' at %1
GS-36305	Expected character to follow '-' at %1
GS-36306	Expected character to follow '[' at %1
GS-36307	Internal error: Called unread() more than once at %1
GS-36308	Internal error: Called unread() more than once at %1
GS-36309	Invalid escape "\\%1" at %2

Table 246: Error messages for the Package util

Number	Description
GS-34200	Internal error: uri is not allowed to be null
GS-34201	Internal error: localName is not allowed to be null
GS-34202	Internal error: uri is not allowed to be null
GS-34203	Internal error: localName is not allowed to be null
GS-34204	Internal error: uri is not allowed to be null
GS-34205	Internal error: localName is not allowed to be null
GS-34206	Internal error: uri is not allowed to be null
GS-34207	Internal error: localName is not allowed to be null
GS-34208	Internal error: uri is not allowed to be null
GS-34209	Internal error: '0'-'3' or 'u' expected after '\\': %1
GS-34210	Internal error: '0'-'7' or 'u' expected after '\\': %1
GS-34213	Unmatched quote at end of string "%1"}

Number	Description
	<p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-34214	<p>Unmatched quote in string "%1" at position%2</p> <p>Tip: This is an <i>attribute error</i>. Attribute errors are displayed in a way that makes them clickable in Genero Studio, as they refer to a particular attribute in the report design document.</p>
GS-34215	failed to map file %1 into memory. Reason: %2
GS-34216	corrupted file %1. Corruption type: 1
GS-34217	corrupted file %1. Corruption type: 2
GS-34218	corrupted file %1. Corruption type: 3
GS-34219	corrupted file %1. Corruption type: 3 at position %2
GS-34220	Internal error: Encountered open Edge
GS-34221	Internal error: Encountered open CharacterEdge

Web Services

Information about using Web Services in Genero Studio.

- [Create a Web Services program](#) on page 910
- [Add Web Service](#) on page 910
- [The Web Services wizard](#) on page 911
- [Build the application](#) on page 913
- [Update the WSDL](#) on page 914
- [Generating web services](#)

See also the Web Services topic in the *Genero Business Development Language User Guide*.

Create a Web Services program

You can use Genero Studio to write, compile, and execute a Genero Web Services (GWS) client or server application.

1. Add an application, library, or folder node to a project in Project Manager.
2. Right-click the node, and select **Add Web Service**. Select either **Client Consuming WSDL** or **Server Implementing WSDL**.

This calls the [The Web Services wizard](#) on page 911, to assist you in adding the Web Services client or server information to the application.

3. Use Code Editor to create or modify a Genero application that calls a Web Service (client) or defines a Web Service (server).

Use the [WSDL information](#) to determine the service name and functions. Code Editor recognizes and parses the syntax of BDL functions related to Genero Web Services. See *the Genero Web Services User Guide* for details on coding your Web services clients and servers.

4. In Genero Studio Project Manager, [add the WSHelper.42m file](#) to the external dependency property of your Genero application; this compiled file contains functions used internally for Web Services.
5. Use the Genero Studio [Build](#) menu option to compile and link your new application.

If you must access the web service using a proxy, provide the proxy information using **Tools >> Preferences**.

Add Web Service

To access or define a Web service, you must have the WSDL description created by the service provider. You can use the WSDL description to determine the operations of the Web Service to use in your BDL program, the required parameters, and the return values.

1. Right-click an **Application**, **folder** or **library node** in Project Manager and select **Add Web Service**. The Web Services Wizard displays.
2. Choose between:
 - a) **Client Consuming WSDL** - to retrieve the WSDL information and generate a . 4gl file containing the BDL functions for the operations in the WSDL, which can be called by a GWS client application.
 - b) **Server Implementing WSDL** - to retrieve the WSDL information and generate a 4gl file (GWS Server Application) containing the BDL functions to create and publish a Web Service which can be accessed over the web.

The [Web Services Wizard](#) guides you through the steps to add the necessary service information to your application.

The Web Services wizard

The Web Services Wizard guides you through the steps to add the necessary service information to your application.

Web Services selection Page

Provide the Service details to get the WSDL description, using one of these:

- **URL** - Provide the URL of the web service. Example: `http://servername:port/serviceName?WSDL`
- **File** - Provide the name and location of a WSDL file in your file system.

Click the **View** button. The description of the Web Service is retrieved from the WSDL and displayed.

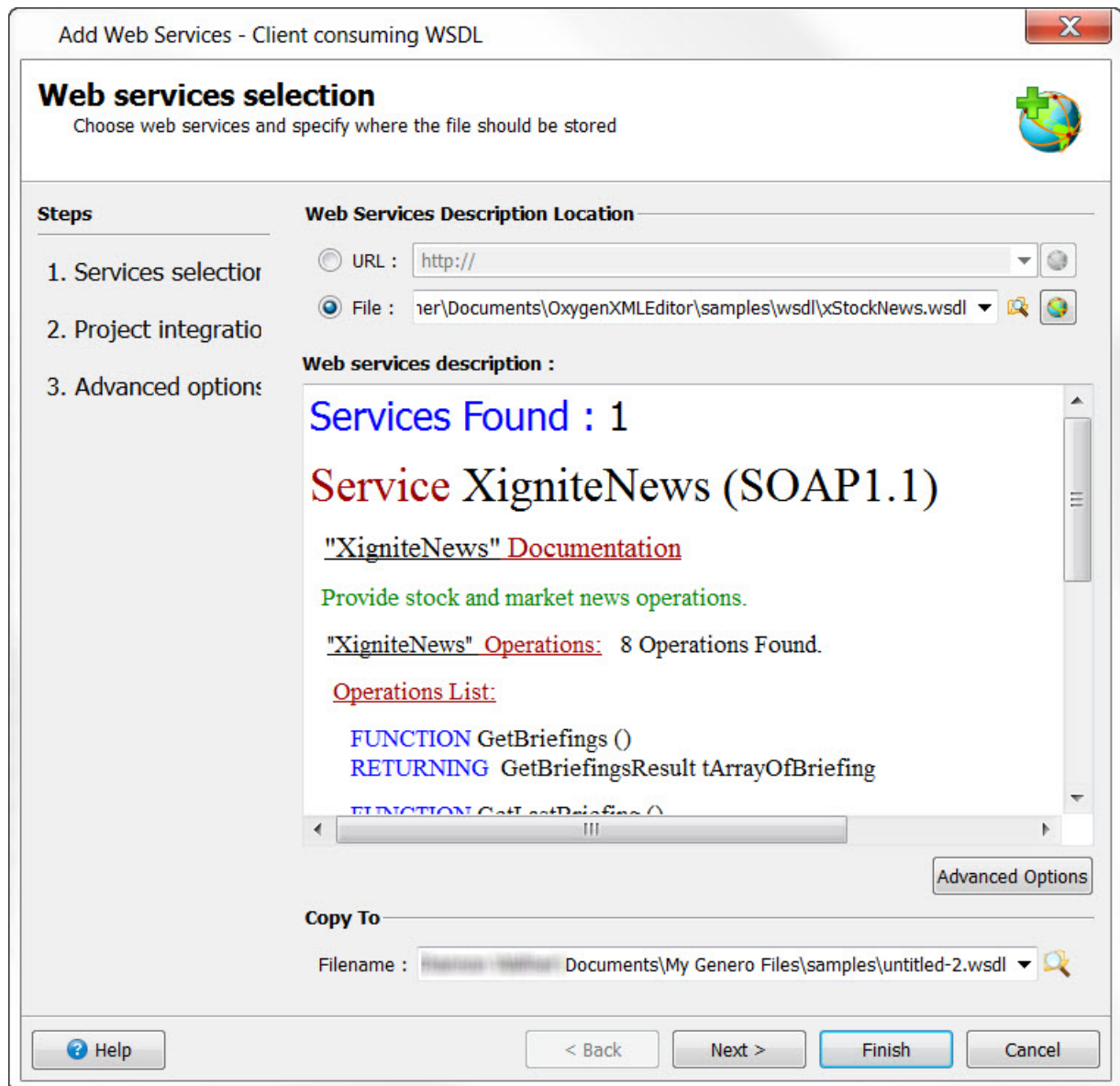


Figure 391: Add Web Services - Client consuming WSDL dialog

Copy to - Use the Browse button to specify the directory where a copy of the WSDL file should be saved. If you used the File option to specify the WSDL file, this directory must be different.

Advanced Options button - if you select this button, a dialog displays that allows you to add an external schema file with the WSDL file.

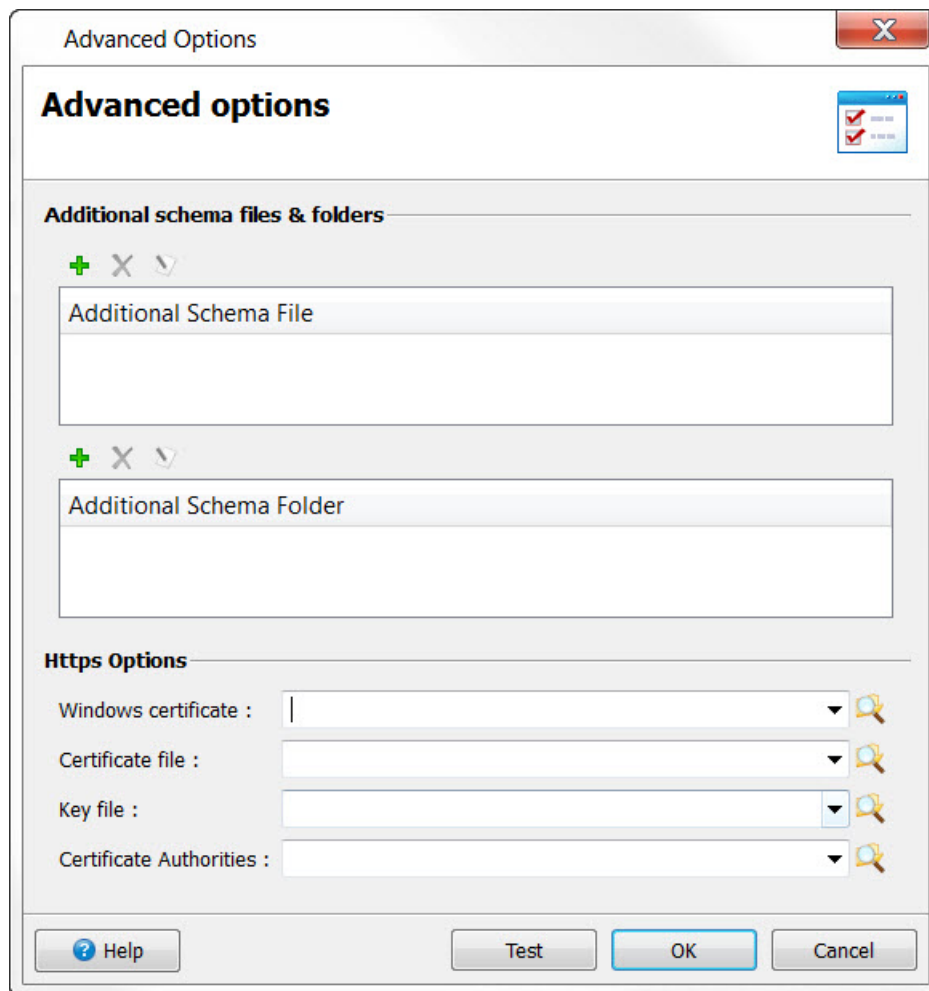


Figure 392: Add Web Services - Client consuming WSDL, Advanced options

Within the Advanced Options dialog:

- **Additional Schema File** - press the **Add** icon to add an external schema file. An Add dialog opens, allowing you to search for the file. Other icons allow you to delete a previously added file, or to edit it.
- **Additional Schema Folder**- press the **Add** icon to add the path and filename of the schema folder.
- **Https Options** - Set the options needed for the fglsdl tool, which implements adding the external schema.

Web Services Project Integration (Client only)

Add Web Services to Project - if this radio button is selected, the WSDL file will be added to your project in Project Manager. Files that show the Web Services operations as functions in a Genero Program are generated and stored in the Target directory of the application:

- `inc` file - the globals file containing the definitions of the input and output records, and the prototypes of the operations.
- `4gl` file - contains the definitions of the functions for your GWS client or server application. and the code that manages the Web Service request or publishes a service,

Advanced - this radio button gives you the flexibility to write your own web client stub. The `4gl` and `inc` files are generated and added to your project in Project manager. The `WSDL` file is created in the "copy to" path.

Important: Do not modify the generated `4gl` files; they are regenerated on each re-build, so modifications will be lost.

Advanced Code Generation Options

Select the options for code generation and runtime behavior, if you wish to modify the default behavior. See The `fglwsdl` tool for a complete list of the command options.

- **Select service/binding** - Choose from the list of all Services or binding in the generated `wsdl` file.
- **Behavior and Functionality**
 - **Compatible with Genero 1.xx** - Implements the "-compatibility" option of the Genero tool `fglwsdl`.
 - **Force RPC convention** (applies to RPC only) - Implements "-fRPC" option of `fglwsdl`.
 - **Generate choice records for inherited types** - Implements "-flInheritance" option of `fglwsdl`.
 - **Ignore SOAP faults** - Implements "-ignoreFaults" option of `fglwsdl`.
 - **Use WS-addressing** - Allows support of WS-Addressing (only WS-Addressing 1.0 is supported.)
 - **Check data type constraints (facets) for simple types** - Implements "-noFacets" option of `fglwsdl`. By default facet is present, so this option is enabled.
 - **Generate XML list of arrays** - Implements "-fArray" option of `fglwsdl`.
 - **Generate DOM callback handlers** - Implements "-domHandler" option of `fglwsdl`.
 - **Generate FGLPROFILE aliases for server URL** Generates `FGLPROFILE` Logical names in place of URLs for the client stub - implements "-alias" option of `fglwsdl`.
- **Coding Conventions & Style**
 - **Include service documentation** - Implements "-comments" option of `fglwsdl`.
 - **Prefix functions, variables, and types with** - Implements "-prefix" option of `fglwsdl`.
 - **Prefix variables and types with namespace** - Implements "-autoNsPrefix" option of `fglwsdl`. Specify the number of words of the namespace that are to be excluded from the prefix. The default prefix is the full namespace, and you may exclude up to nine words. After the ninth word, you can only add the last word of the namespace as the prefix.
 - **Define prefix for variables and types per namespace** - Implements "- nsPrefix" option of `fglwsdl`. For each namespace, you can define a prefix for variables and types.
- **Command Line** - additional arguments for the execution of the `fglwsdl` tool.

The command line is read-only, and is automatically updated when you select an option.

Code Generation Result - optional page

This page displays only if you have selected **Advanced** in the [Web Services Project Integration](#) page of the wizard.

Select **Finish** to generate the `WSDL` file and accompanying files.

Note: If you must access the web service using a proxy, provide the proxy information in [General Preferences](#) on page 106.

Build the application

The Genero library file `WSHelper.42m` must be linked into every application using Web Services.

[Build Rules](#) are defined to link the generated `4gl` file into your application. The generated `4gl` file in the Target Directory should not be modified, as your changes will be lost if the file is regenerated.

Important: The Genero library file `WSHelper.42m` must be linked into every application using Web Services. Add the library file to the application node in Project Manager as an external dependency property, prior to building the application.

When you build the Project, the appropriate files are compiled and linked.

Update the WSDL

Right-click the WSDL node in the project structure to update the WSDL file.

An internet connection is required, unless the Web Services server is local.

While updating the file, any new dependency files are also downloaded.

Add Web services (Server, Services, Forms with services)

Web services are a standard way of communicating between applications over an intranet or Internet. They define how to communicate between two entities: a server that exposes services and a client that consumes services. Web services may provide programmable access to the functions of a form or a standalone service without a form.

- [Webservice Server entity](#) on page 216
- [Create a Webservice server](#) on page 216
- [Webservice entity](#) on page 217
- [Create standalone service](#) on page 219
- [Create service from a form](#) on page 219
- [CRUD form and Webservice, Zoom form and Webservice](#) on page 219
- [JSON Web services](#) on page 220
- [Public fields](#) on page 220

Mobile applications

Genero Studio provides the framework for creating robust Genero mobile applications.

Mobile development environment

A complete and inclusive development environment allows you to write and test your Genero mobile apps from your desktop prior to deploying them to your mobile devices.

The mobile development environment includes the runtime DVM, the development IDE (Genero Studio), and two development agents: Genero Mobile for iOS (GMI) and Genero Mobile for Android (GMA). During development, the DVM and the IDE sit on the desktop, and the GMI and GMA display clients sit on mobile devices. A mobile device can be a physical device, such as a phone or tablet, or it can be an emulator.

During development, you run the app in *developer mode*. In developer mode, the app runs on a development machine and displays the user interface (UI) to an Android or iOS device or emulator. The server app makes the initial connection. The Genero Mobile Development Client runs on the device, listening at port 6400 (by default) for the connection.

By running the app from your development environment, you can develop and debug your app from within the Genero Studio IDE. You do not have to continuously copy files to the device. You can quickly see the user interface and engage with the app. You can quickly switch between various devices.

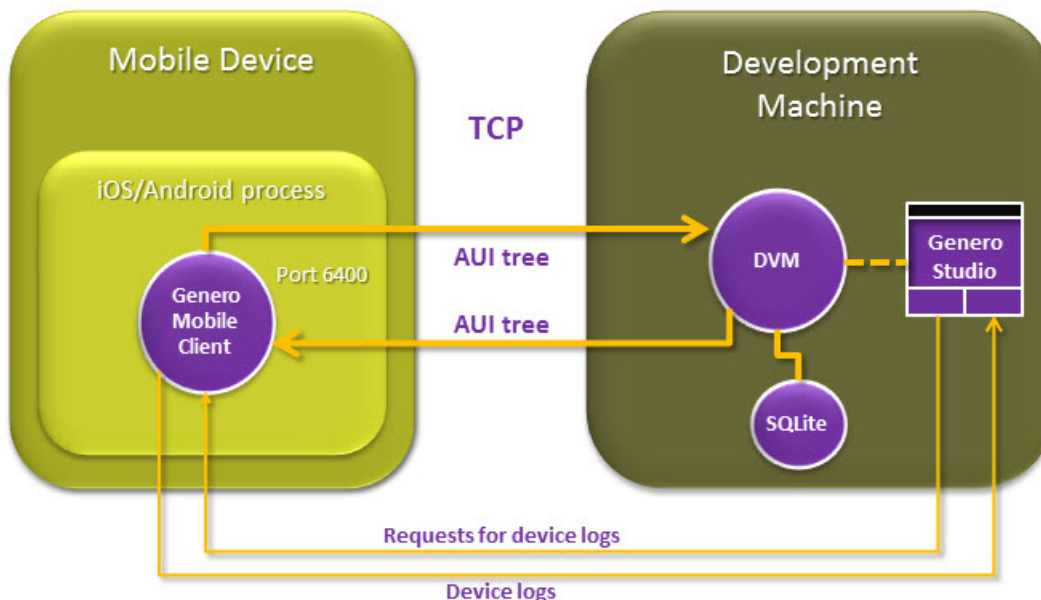


Figure 393: Developer mode

Options for development

Three clients are available for use while developing Genero mobile apps.

- Genero Mobile for Android (GMA)

This client allows you to test your app on an Android physical device or emulator during development.

To put the GMA client on your Android device, you must connect your device to your computer via the USB port.

To run your app in developer mode and view it on your Android device, you must connect your device to your development machine via the USB port.

See [Configure Genero Mobile for Android](#) on page 118 for details on setting up the GMA for development.

- Genero Mobile for iOS (GMI)

This client allows you to test your app on an iOS physical device or simulator during development. The development machine must use a Mac OS.

To put the GMI client on your iOS device, you must connect your device to your computer via the USB port.

To run your app in developer mode and view it on your iOS device, connect to your development machine via the USB port.

See [Configure Genero Mobile for iOS](#) on page 127 for details on setting up the GMI for development.

- Genero Development Client

Using the Genero Development Client, you can view your app on an iOS device when developing with a Windows or Linux development machine. Mac OS is required on the mobile device.

Important: While this client allows you to view and test your app on the device in developer mode, it does not give you the ability to create an iOS package or to deploy your app to an iOS device.

To put the Genero Development Client on your iOS device, you must download the app from the Apple store.

To run your app in developer mode and view it on your iOS device using the Genero Development Client, you must use a wireless network (wifi).

See [Display to the Genero Mobile Development Client](#) on page 138 for details on using the Genero Development Client.

Tools to assist with development

Genero Mobile has a variety of tools to aide in development.

- Preview the form on the mobile device.
- Debug your app using the graphical debugger.
- Use the profiler to gather statistics on where your program spends its time and to identify bottleneck functions.
- View the AUI tree in a browser on your development machine (or any browser on the local wifi network).
- Display the program logs.
- Display the device logs.

See [Graphical Debugger](#) on page 502 for deatils on the graphical debugger. The command-line debugger (fgldb) is described in the *Genero Business Development Language User Guide*.

See [Debugging a mobile app](#) on page 918.

Using command-line tools

Working from the desktop, you have the full suite of command line tools at your disposal: `gsmake`, `fglrun`, `fglrun -d`, and so on. See the *Genero Business Development Language User Guide* for information on the command line tools.

Deploy to the device for testing

The development clients are provided to ease your development efforts, allowing you to view your app without having to package and deploy your app to the device to view changes in the app code. Towards the end of the development cycle, you will want to test your app running fully on the device.

When a package is deployed on a device where the debug package has been activated, debugging tools available include:

- View the AUI tree in a browser.
- Display the program logs.
- Display the device logs.
- Command-line on-device debugging.

See [Packaging, deploying, and distributing apps](#) on page 993.

Genero mobile app demos

There are several demos provided to highlight Genero mobile apps.

If you are looking for code snippets and examples, you are encouraged to view the demo apps that have been created to highlight Genero mobile apps. You can run the demos in developer mode or you can deploy each demo to your device.

OfficeStoreMobile

OfficeStoreMobile is provided as one of the default projects. Access the project from the Tutorials & Samples tab on the Welcome Page. This app was created using the Business Application Modeler.

This demo is the Office Store demo, where you can view a list of customers and their orders.

You can deploy the package for the app to your device or emulator. The package can be found under the Packages node in the project.

MobilePatterns

MobilePatterns is provided as one of the default projects. Access the project from the Tutorials & Samples tab on the Welcome Page. This app was created using the Business Application Modeler.

This demo shows how you can define relationships between two forms, and how you can use a common form with multiple relationships defined.

You can deploy the package for the app to your device or emulator. The package can be found under the Packages node in the project.

PhoneDemo

The PhoneDemo includes the [Stores demo](#) and additional demos that show how various forms, menus and widgets are displayed on your mobile device. You can access the project from the Tutorials & Samples tab on the Welcome Page.

The source code can be found at `$GSTDIR/samples/PhoneDemo` or, if you are not using Genero Studio, `$FGLDIR/demo/MobileDemo/phoneMain`.

In addition to running the demo in developer mode or deploying the demo to your device, you can start the PhoneDemo from the Genero Mobile client.

Table 247: Starting the PhoneDemo from your Genero Mobile client

GMA	GMI
<ol style="list-style-type: none"> 1. Start Genero Mobile on your device. 2. Tap Browse bundled apps. 3. Tap InternalGeneroApps. 4. Tap MobileDemo.xcf. 	<ol style="list-style-type: none"> 1. Start Genero Mobile on your device. 2. Tap Run Demo.

The Stores demo is a non-generated app that provides a simple order tracking app for a fictional sporting goods store.

The source code for the Stores Demo can be found at `$GSTDIR/samples/PhoneDemo/stores` or, if you are not using Genero Studio, `$FGLDIR/demo/MobileDemo/stores`.

stores2

The Stores2 demo is the Stores demo written for a tablet device. It demonstrates the implementation of the split view and the navigator pane. You can access the project from the Tutorials & Samples tab on the Welcome Page.

The source code can be found at `$GSTDIR/samples/PhoneDemo/stores2` or, if you are not using Genero Studio, `$FGLDIR/demo/MobileDemo/stores2`.

stores-server

The stores-server demo contains the daemon to sync the stores database on the server.

The source code can be found at `$GSTDIR/samples/PhoneDemo/stores-server` or, if you are not using Genero Studio, `$FGLDIR/demo/MobileDemo/stores-server`.

Debugging a mobile app

You have several tools available for debugging your Genero Mobile application.

To use the debugging tools, you will either have:

- An app running in developer mode, with the Debug service enabled.
- A debug version of a deployed app.

Topics:

- [Debug version of a deployed app](#) on page 919
- [Debug tools for apps in developer mode](#) on page 920
- [Viewing the AUI tree](#) on page 921
- [Viewing the program logs](#) on page 923

- [Viewing the device logs](#) on page 924
- [Debugging a Web Component](#) on page 925

Debug version of a deployed app

Debugging methods and tools are available for the debug versions of standalone apps.

- [Create a debug version of a deployed app](#) on page 919
- [Run the debug version of a deployed app \(Android\)](#) on page 919
- [Run a debug version of a deployed app \(iOS\)](#) on page 920

Create a debug version of a deployed app

To create a debug version of an app, you must set the `DEBUG_PACKAGE` environment variable to 1 (or `TRUE`) prior to packaging the app.

Before you begin, your app project must be open in Genero Studio.

Tip: When using the `fgldb` command line utility to debug a standalone app, you may need to alter your application in order to set a breakpoint in the `MAIN` module. This is necessary because the app needs to be running and interactive to start the use of `fgldb`. To enable this, you would start your app with this code:

```
MAIN
  MENU
    COMMAND "exit"
    EXIT MENU
  END MENU
  . . .
END MAIN
```

With this addition, one can connect easily using `fgldb` from the command line. For details on using `fgldb`, see the *Genero Business Development Language User Guide*.

1. Select **Tools >> Genero Configurations**.

2. Select a Configuration Name.

The current configuration is selected by default.

3. Under **Environment Sets**, check the **Mobile Debug Package** environment set.

Within this environment set, the `DEBUG_PACKAGE` environment variable is set to 1 (`TRUE`). to build a debug version of the package.

4. Package the app. See [Package a mobile app](#) on page 996.

The package for the debug version of the app is created and ready to be deployed.

Run the debug version of a deployed app (Android)

After deploying the app to your device, you must start the app in order to use the debug tools.

Before you begin, the debug version of the app must be deployed to your device or emulator.

Note: You must connect the physical device to your computer using a USB cable in order to use any debug tools with a deployed debug application.

1. On your device, launch your app.

2. If the debug service is running on the device, stop it.

If the debug service is running, the red Genero Mobile debug service icon appears in the status bar.

a) Drag down from the top of the screen to display a menu.

b) Below the Genero Mobile debug service entry, tap **Cancel**.

The red Genero Mobile debug service icon no longer appears in the status bar.

3. Set **Debug service** to **ON**.

The red Genero Mobile debug service icon appears in the status bar.

4. Tap **Browse bundled apps**.
5. Tap **InternalGeneroApps**.
6. In the list, find the two entries for your app.

The two entries are file names: *programname.42r* and *packagenodename.xcf*.

Note: If the package node name and the program name are identical, only one entry (*programname.xcf*) displays.

7. Tap on either file name to start the app.
The app starts in debug mode. You can view the AUI tree or log files, or you can connect to and debug your application using the `fgldb` command line utility.

Run a debug version of a deployed app (iOS)

After deploying the app to your device, you must start the app in order to use the debug tools.

Before you begin, the debug version of the app must be deployed to your device or emulator.

Note: You must connect the physical device to your computer using a USB cable in order to use any debug tools with a deployed debug application.

With Genero Mobile for iOS, the debug version of a deployed app enables the debug preferences by enabling port 6400 for debugging.

Open the app.

The app starts in debug mode. You can view the AUI tree or log files, or you can connect to and debug your application using the `fgldb` command line utility.

Debug tools for apps in developer mode

Some debug tools are only available when running your app in developer mode.

These debug tools are only available for apps running in developer mode.

- [Preview a form on a mobile device](#) on page 920
- [Run an app with the graphical debugger](#) on page 920
- [Run an app with the Profiler](#) on page 921

These debug tools can be used for apps running in developer mode.

- [View the AUI tree \(Android\)](#) on page 921
- [View program logs \(Android\)](#) on page 923
- [View device logs \(Android\)](#) on page 924

Preview a form on a mobile device

When working in developer mode, you can preview your form on a mobile device.

Before you begin, select the appropriate Genero Studio configuration for displaying to your mobile device or emulator, and have your app project open in Genero Studio.

1. Open a form specification file (`4fd` or `4fdm`).
2. Select **Build >> Preview**.

The form displays on your device.

Run an app with the graphical debugger

When working in developer mode, you can use the graphical debugger to step through your code as your app executes on your mobile device.

Before you begin, select the appropriate Genero Studio configuration for displaying to your mobile device or emulator, and have your app project open in Genero Studio.

1. Open an app source (`4g1`) file.
2. In the source (`4g1`) file, use **Debug >> Add/Delete Breakpoint** to set your breakpoints.

3. Select **Debug >> Debug**.

The app starts on your device.

4. Use the graphical debugger icons to step through your code.

For more information on using the graphical debugger, see the *Genero Studio User Guide*.

Run an app with the Profiler

The Profiler is a tool built in the runtime system that generates a report about where the program spends time, and which function calls which function. The Profiler can help to identify areas in the program that are slower than expected.

Before you begin, select the appropriate Genero Studio configuration for displaying to your mobile device or emulator, and have your app project open in Genero Studio.

For more information on using the Profiler, see the *Genero Studio User Guide*.

Note: When in developer mode, the DVM is running on the desktop. The Profiler is therefore providing a profile of the app as it runs on the desktop. It may not be a true representation of how it will perform when deployed to the device.

1. Set the desired app as the default app.

2. Select **Debug >> Execute with Profiler**.

The app starts on your device.

3. Use the app.

4. End the app.

The Profiler report appears in the Output panel.

Viewing the AUI tree

You can view the AUI tree in a browser. The AUI tree is the tree of the program's objects and user interface elements, and can be used to see what is present in the AUI tree for the program and the values of each element's properties.

The app must be running in developer mode or be the debug version of a deployed app.

- [View the AUI tree \(Android\)](#) on page 921
- [View the AUI tree \(iOS\)](#) on page 922
- [View the AUI tree \(Development Client\)](#) on page 923

View the AUI tree (Android)

You can view the AUI tree for your Android mobile app in a browser.

Before you begin, your app must be able to run in development mode or be the deployed debug version of the app. You must connect the physical device to your computer using a USB cable in order to use any debug tools with a deployed debug application.

1. If you are going to run your app in developer mode, complete these steps.

a) From Genero Studio, launch your application.

b) Select **Tools >> Android Tools >> Show AUI Tree**.

The AUI Tree displays in a browser.

c) Click on a node in the tree to highlight the corresponding item on the device or emulator.

2. To view your app in a browser while running the deployed debug version of your app using Genero Studio, complete these steps.

a) To run the app from your device, launch your app in debug mode. See [Run the debug version of a deployed app \(Android\)](#) on page 919.

The application launches and the initial form displays.

b) If you are using Genero Studio, select **Tools >> Android Tools >> Show AUI Tree**.

The AUI Tree displays in a browser.

c) Click on a node in the tree to highlight the corresponding item on the device or emulator.

3. To view your app in a browser while running the deployed debug version of your app, complete these steps.

a) To run the app from your device, launch your app in debug mode. See [Run the debug version of a deployed app \(Android\)](#) on page 919.

The application launches and the initial form displays.

b) Swipe down on the icon bar and locate the entry for the Debug service. Make a note of the URI address provided.

The URI takes the form of `http://<device_ip_address>:<port_number>`.

c) Open a browser and enter in the URI from the previous step.

d) Click **AUI tree**.

The AUI Tree displays in a browser.

e) Click on a node in the tree to highlight the corresponding item on the device or emulator.

Important: The AUI tree does not automatically update in your browser as you move around the app. As you navigate the app on your device, you must reload your browser to view the changes in the AUI tree.

View the AUI tree (iOS)

You can view the AUI tree for your iOS mobile app in a browser.

Before you begin, your app must be able to run in development mode or be the deployed debug version of the app. You must connect the physical device to your computer using a USB cable in order to use any debug tools with a deployed debug application.

1. If you are going to run your app in developer mode, complete these steps.

a) From Genero Studio, launch your application.

b) Select **Tools >> iOS Tools >> Show AUI Tree**.

The AUI Tree displays in a browser.

c) Click on a node in the tree to highlight the corresponding item on the device or simulator.

2. If you are going to run the deployed debug version of your app, complete these steps.

a) On the iOS device, open Genero Mobile.

b) In the **INFORMATION** section, note the **HTTP URL** provided.

For example, HTTP URL=`http://192.168.0.160:6400`

c) Launch your application in debug mode. See [Run a debug version of a deployed app \(iOS\)](#) on page 920.

d) Swipe down to view details about the icons in the status bar.

The description for the Genero Mobile debug service displays the URL you must use to access debug details. It takes the form `http://device-ip-address:port-number`.

e) Enter the URL in a browser.

f) Click **AUI tree**.

The AUI Tree displays in a browser.

g) Click on a node in the tree to highlight the corresponding item on the device or simulator.

3. If you are going to run the deployed debug version of your app, and your machine (running Genero Mobile) and device (running the app) are on the same wireless network, you can use Genero Studio to view the AUI tree.

a) On the iOS device, open Genero Mobile.

b) Select **Tools >> Genero Configurations**.

c) Select your iOS configuration from the list.

d) Click the Edit icon next to the **Use Display Client** field.

The Display Client management window opens.

e) In the **Host** field, enter the ip address of your device. Click **OK** until all configuration management windows close.

- f) Launch your application in debug mode. See [Run a debug version of a deployed app \(iOS\)](#) on page 920.
- g) From Genero Studio, select **Tools >> iOS Tools >> Show AUI Tree**.
The AUI Tree displays in a browser.
- h) Click on a node in the tree to highlight the corresponding item on the device or simulator.

Important: The AUI tree does not automatically update in your browser as you move around the app. As you navigate the app on your device, you must reload your browser to view the changes in the AUI tree.

View the AUI tree (Development Client)

While viewing your app in the Genero Development Client, you can view the AUI tree for your mobile app in a browser.

Before you begin, you should have verified your configuration within Genero Studio to display to the Genero Development Client.

The Genero Development Client allows the developer to view the AUI tree on the default port 6400.

1. On your iOS device, launch the Genero Development Client.
2. Find the **HTTP VIEWING URL**. Make a note of the URL.

Note: The **HTTP VIEWING URL** displays in the Development window. If you do not see the development window, press the **Develop** button.

3. In Genero Studio, set your configuration for the **iOS Dev Client** and start your app.
In the Task pane, your app shows as running.
4. In the Genero Development Client interface on your iOS device, click **Connect**.
The app displays on the device.
5. Open a browser and enter in the HTTP VIEWING URL.
The GMI Information page displays.
6. Click **AUI Tree**.
The AUI Tree displays.

As you navigate the app on your device, you must reload your browser to view the changes in the AUI tree.

Viewing the program logs

You can view the program logs in a browser.

- [View program logs \(Android\)](#) on page 923
- [View program logs \(iOS\)](#) on page 924
- [View program logs \(Development Client\)](#) on page 924

View program logs (Android)

You can view the program logs for your Android mobile app in a browser. The output displays VM messages (standard output and standard error).

Before you begin, your app must be the deployed debug version of the app. You cannot view the standard output and error logs when running an app in developer mode. You must connect the physical device to your computer using a USB cable in order to use any debug tools with a deployed debug application.

1. To view the standard output and error logs in a web browser:
 - a) To run the app from your device, launch your app in debug mode. See [Run the debug version of a deployed app \(Android\)](#) on page 919.
The application launches and the initial form displays.
 - b) Swipe down on the icon bar and locate the entry for the Debug service. Make a note of the URI address provided.
The URI takes the form of `http://<device_ip_address>:<port_number>`.
 - c) Open a browser and enter in the URI from the previous step.

- d) Click **VM Output**.

The standard output and standard error messages display.

2. To view using the **Display Standard output and error** menu option:

- a) To run the app from your device, launch your app in debug mode. See [Run the debug version of a deployed app \(Android\)](#) on page 919.

The application launches and the initial form displays.

- b) Select **Tools >> Android Tools >> Display Standard output and error**.

The program logs (standard output and standard error messages) are written to the Output view.

- c) To stop the program logs from appearing in the Output view, select **Tools >> Android Tools >> Stop display Standard output and error**.

View program logs (iOS)

You can view the program logs for your iOS mobile app in a browser.

Before you begin, your app must be able to run in development mode or be the deployed debug version of the app. You must connect the physical device to your computer using a USB cable in order to use any debug tools with a deployed debug application.

1. On the iOS device, select **Settings >> Genero Mobile**.
2. Enable GUI LOGGING.
3. Open Genero Mobile (GMI).
4. In the **INFORMATION** section, note the **HTTP URL** provided.
For example, HTTP URL=`http://192.168.0.160:6400`
5. On your development machine, open your project in Genero Studio.
6. Start your app.
7. Open a browser and enter the HTTP URL identified in a previous step.
8. Click **Logs**.
The program logs display in the browser.

View program logs (Development Client)

While viewing your app in the Genero Development Client, you cannot view the program logs in a browser.

The Genero Development Client acts as a remote GUI client, and therefore does not have logging options for the client. In order to get logs for an app displayed in the Genero Development Client, you need to enable the virtual machine's `--start-guilog` option when starting the app.

See *Front-end protocol logging* in the *Genero Business Development Language User Guide* for details regarding the `--start-guilog` option.

Viewing the device logs

You can view the device logs in a browser.

The app must be running in developer mode or be the debug version of a deployed app.

Note: At this time, only Genero Mobile for Android offers this feature.

- [View device logs \(Android\)](#) on page 924

View device logs (Android)

You can view the Android device logs in a browser on your development machine.

Before you begin, your app must be able to run in development mode or be the deployed debug version of the app. You must connect the physical device to your computer using a USB cable in order to use any debug tools with a deployed debug application.

1. If you are going to run your app in developer mode, complete these steps.
 - a) From Genero Studio, launch your app.

- b) Select **Tools >> Android Tools >> Show Device Logs**.
The device logs display in the Output view. New log entries will be appended in real time.
2. If you are going to run the deployed debug version of your app, complete these steps.
 - a) To run the app from your device, launch your app in debug mode. See [Run the debug version of a deployed app \(Android\)](#) on page 919.
The application launches and the initial form displays.
 - b) Swipe down on the icon bar and locate the entry for the Debug service. Make a note of the URI address provided.
The URI takes the form of `http://<device_ip_address>:<port_number>`.
 - c) Open a browser and enter in the URI from the previous step.
 - d) Click **Android Logcat**.
The device logs display.

Debugging a Web Component

You can debug a web component on a Genero Mobile application.

- [Debug a web component on an Android device](#) on page 925
- [Debug a web component on an iOS device](#) on page 925

Debug a web component on an Android device

You can enable the debugging of a Web component through Chrome.

This procedure requires:

- GMA 1.1 or greater.
- Android 4.4 or greater.

You need enable the GMA debug mode and follow the instructions provided at <https://developer.chrome.com/devtools/docs/remote-debugging#configure-webview>

1. Connect your device to your desktop with a USB cable.
2. If GMA is running on your device, stop GMA on your device.
3. Enable USB debugging on the device.
4. Run your application with a web component.

Tip: Please be patient as the application may take longer to start.

GMA starts with debug service enabled.

5. Open Chrome on your desktop.
6. In the Chrome address bar, enter `chrome://inspect/#devices`.
7. Check **Discover USB devices**.
8. Accept the USB debugging on your device.

A list of URLs appear. You can inspect the individual pages.

Debug a web component on an iOS device

This configuration allows you to debug a web component on an iOS device.

For GMI, you can only debug with the iOS simulator. You cannot debug on the physical device.

1. Start the app.
2. Navigate to the webview you want to debug.
3. Open Safari.
4. Make sure the **Show Develop menu in toolbar** is enabled in **Preferences >> Advanced**.
5. Go to **Develop >> iOS Simulator** and choose the html to debug.

Localize a mobile app

Follow this procedure to localize your mobile app.

Before completing this procedure, you have an app that works but is not yet localized.

You can use localized string files to provide different languages for your mobile app. Before you complete the steps in this procedure, it would benefit you to read on the use of localized strings in the *Genero Business Development Language User Guide*.

Important: An app must be deployed to see localization take place. Localization does not work in developer mode.

1. Modify your app source and resource files for localized string use.
 - a) For your text-based source files (`4g1` and `per`), see the article *Localized strings in program sources* in the *Genero Business Development Language User Guide*.
 - b) For your graphical form files (`4fd`), see [Localizing your form](#) on page 443.
 - c) For your resource files (such as your action defaults (`4ad`) and presentation styles (`4st`) files), see the article *Localized strings in XML resource files* in the *Genero Business Development Language User Guide*.
2. Create the localized string files.
 - a) For your text-based source files (`4g1` and `per`), see the article *Extracting strings from sources* in the *Genero Business Development Language User Guide*.
 - b) For your graphical form files (`4fd`), see [Localizing your form](#) on page 443.
 - c) For your resource files (such as your action defaults (`4ad`) and presentation styles (`4st`) files), an extraction tool is not provided. You must create the localized string file by hand.

You now have a localized string file for each of your source files. These files serve as the starting point for language-specific localized string files; you will create a copy these files for each language you wish to provide.

Tip: Save these base localized string files in your Resources folder in your project. You do not, however, want them to be compiled. Select each of the base localized string files and check the Exclude from compilation property.

3. Update your `fglprofile` file to list the required string files.
See *Using localized strings at runtime* in the *Genero Business Development Language User Guide*.
4. Identify the languages and their corresponding locale codes needed for your localization effort.
The selected language is identified by a locale code following the ISO 639 standard. See *Using localized strings at runtime* in the *Genero Business Development Language User Guide*.
5. Create a sub-directory for each language you wish to localize.
For the directory name, use the locale code identified in the previous step.
6. Copy all the localized string files into each locale sub-directory.
7. Edit the localized string files within each locale sub-directory to provide the translations. Save your changes.
8. Compile each localized string file.

If you are using the command line to compile the localized string files, see *Compiling string files* in the *Genero Business Development Language User Guide*. If you are using Genero Studio to compile the localized string files, follow these steps.

- a) Create a library node for each language.
For example, you could create a library named **en** to hold your generic English localized string files, and you could create a library named **en_US** to hold your English (United States) localized string files.
- b) Select the library node and change the **Target directory** property to `$(ProjectDir)/bin/xx`, where `xx` is the language code.

For example, you would specify `$(ProjectDir)/bin/en` for the generic English library node, or `$(ProjectDir)/bin/en_US` for the English (United States) library node. The target directory specifies where the compiled string files will be placed after compilation. These examples assume that the compiled program files are being placed in `$(ProjectDir)/bin`.

- c) Right-click on the library node and select **Add Files** to add your localized string files (`str`) from the corresponding language directory on disk.
 - d) Right-click on the library node and select **Build**.
The compiled localized string files (`42s`) appear in the specified Target directory.
 - e) Right-click on the **Application** node and select **Advanced Properties**.
 - f) Select the **Dependencies** page and select each language library in the **Dependencies** property page.
9. For each package node, add a Directory node for each language library node.
The Directory node places the compiled string files into the correct directory. For example, if the **Source directory** is set to `$(ProjectDir)\bin\en_US`, then the **Destination directory** must be set to `$(ProjectDir)\bin\en_US`. Set the **Included files** filter to `*.42?`.
10. Deploy your package.
11. Test each language.
- a) Change the language in the device's settings to one of the languages you have set for localization.
 - b) Launch your app.

Genero Mobile error messages

A list of Genero Mobile error messages. For messages that are not self-explanatory, additional information is provided.

Table 248: Genero Mobile error messages

Number	Description
GS-09002	File path issue. Path is incorrect or file is missing. Cannot package file : \$filePath source and destination relative path don't match Destination directory not in rootDir Cannot compute zip directory from source dir Error updating application archive '\$appZip' Error compressing application archive '\$appZip' Error adding \$packageName\.xcf to application archive '\$appZip'
GS-09003	Bad script arguments such as an incorrect build/package/deploy/run rule. Missing argument apkFilePath Missing argument deployData Missing argument projectDir Missing argument architecture
GS-09004	Genero configuration settings errors. Missing dynamic property '\$propertyName' Missing environment variable GMADIR

Number	Description
	<p>Cannot find Genero Mobile for Android package, check GMADIR environment variable</p> <p>Missing environment variable ANDROID_HOME</p> <p>Provisioning profile '\$provisioningProfile' doesn't exist: The path contained in PROVISIONING_PROFILE variable is not valid.</p> <p>Provisioning profile '\$provisioningProfile' is not a provisioning profile: The path contained in PROVISIONING_PROFILE variable is not a provisioning profile (.mobileprovision) file.</p> <p>Copy of provisioning profile '\$provisioningProfile' failed: The copy of provisioning profile to package has failed.</p> <p>Invalid value for DEBUG_PACKAGE variable (should be an integer superior or equal to 0).":DEBUG_PACKAGE value is invalid. It should contains the debug level (a positive integer).</p> <p>Cannot find Genero Mobile for iOS package, check GMIDIR environment variable.: Path contained in GMIDIR is not valid.</p>
GS-09005	Cannot read deployment data file '\$deployData'
GS-09006	Error parsing deployment data file '\$deployData'
GS-09007	<p>Informational messages.</p> <p>Deploy to \$deviceId started</p> <p>Deploy started</p> <p>Deploy finished</p> <p>Starting emulator</p> <p>Waiting for \$deviceId to respond</p> <p>Emulator started</p> <p>Starting Genero Mobile for Android on \$deviceId</p> <p>Forwarding android port 6400 to localhost:\$displayClientPort</p> <p>Starting Genero Mobile for Android Genero Mobile for Android started</p>
GS-09008	<p>Errors in package node in project file (4pw).</p> <p>Invalid package name '\$packageName': Use alpha numeric characters only.</p> <p>Invalid package ID '\$packageId': Follows java package naming: <i>com.company.product</i></p> <p>Invalid packageLabel '\$packageLabel': Package label cannot be an empty string.</p> <p>Invalid package version '\$packageVersion': Package version must be integer value.</p>
GS-09009	No module (.42r) in package
GS-09010	<p>An external process failed.</p> <p>Error displaying \$url in browser : \$msg</p>

Number	Description
	<p>Error forwarding port to \$port : \$errmsg</p> <p>Error launching internet browser \$browser : \$errmsg</p> <p>Error launching adb devices Connect an Android device or start the Android Virtual Device emulator</p> <p>Error decompressing '\$gmaZip' : \$errmsg</p> <p>More than one device and emulator</p> <p>Error launching web browser \$browser : \$errmsg</p> <p>No device connected</p> <p>Several devices connected. Enter device ID in display client configuration</p> <p>Failed to open \$sourceFileName</p> <p>Error while signing 'payload/GMI.app': \$errmsg</p> <p>Error while creating ipa file: \$errmsg</p>
GS-09011	Multiple modules (.42r) in package \$packageName
GS-09012	Incorrect Icon-mdpi extension, only png image format is supported.

BAM Template Developer Guide

Genero Studio comes with a set of templates used with the Business Application Modeler (BAM) for the code generation. You are free to customize these templates or create your own templates so that the generated application meets your requirements.

You can customize and configure Genero Studio and the Business Application Modeler to generate the application you want to build, the way you want to build it.

For example you may find that instead of adding POINT or BLOCK code repeatedly for each application, you want to use a template that generates your customized code. Or, perhaps you want to add custom properties to a modeling diagram.

Options for modifying templates or creating new templates can be categorized in this order of complexity.

Table 249: Options for modifying templates

Option	Types of modifications
Modify a generated application without touching templates.	<ul style="list-style-type: none"> • Modify diagram property values. • Use POINT/BLOCKS to modify generated code. See Using POINTs and BLOCKs on page 262. • Use resource files (styles, action defaults). See Modify action defaults (dbapp.4ad) on page 266, Modify styles (dbapp.4st) on page 267, Modify the Topmenu (dbapp.4tm) on page 267, Modify the Toolbar (dbapp.4tb) on page 267
Modify the default template files.	<ul style="list-style-type: none"> • Add, remove, or change diagram properties • Change generated 4gl code
Customize template files.	<ul style="list-style-type: none"> • Change build rules • Create new file types • Create new File >> New options • Customize Genero Studio diagrams. See Example 1: Adding a new property on page 931, Example 3: Adding an entity to the BA Diagram on page 936 • File templates, code templates, user action

Quick Start: Customizing templates

This Quick Start uses examples to illustrate how to customize template files.

Modifying the default templates requires a basic knowledge of Genero Business Development Language and the Tcl language or another template language. Genero Application Generator uses the Tcl Generator by default; it can be replaced by a generator of your choice such as XSLT or perl.

Example 1: Adding a new property

In this example, the BA diagram is modified to include a new property. When a value is set in the new property, the modified tcl script generates the code to include the information from the property.

Configure a copy of the default template set

Create a new template set from which applications can be generated.

1. Copy the entire content of the `GSTDIR/gst/bin/src/ag/tpl/dbapp` directory to a new directory. The new template directory could be a common one on the server or it could be part of the entire versioned project.
You may also create a new empty directory to which you can add the template files one at a time as you create them.
2. Determine which configuration you want to use to run your generated applications. [Select the configuration](#). Then, select **Tools >> Genero Configurations**.
3. Find and select the **Template** Environment Set.
4. Make a copy of the **Template** environment set using the **duplicate** action in the integrated Toolbar.
5. Rename your template environment if you wish by right-clicking on it and selecting **Rename**.
6. Select your template environment set. Notice that there are environment variables set.
7. Modify `GSTSETUPDIR`. Double-click `GSTSETUPDIR` in the list and modify the Value to point to the location of your template files.
8. Select OK.
9. In the Environment Sets list of Genero configuration management, uncheck the environment set for **Template** and check your environment set.

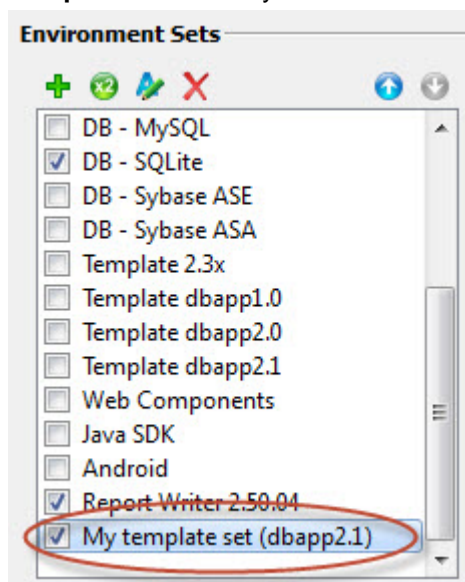


Figure 394: Select new environment set

10. Now you can customize the files that you copied into the new directory. Keep the same file names as the default template set. All your programs will be generated with these files; the default files will be ignored.

Add a new property to the BA diagram

This example illustrates how to add a new property to the Business Application diagram by adding a new icon property to the diagram template.

The screenshot shows the Genero Studio interface. On the left, a Business Application diagram is displayed with several entities and their relationships. The 'Account' entity is highlighted with a red dashed box. On the right, the Properties window is open, showing the properties for the selected 'Account' entity. The 'Miscellaneous' section is expanded, and a new property named 'Icon' with the value 'smiley' is visible, circled in red.

Name	Value
Object	
Name	Account
Type	Program
File Name	C:\Users\Shannon W...
Miscellaneous	
Icon	smiley

Figure 395: New property in Program entity properties

1. From Genero Studio, open the `settings.agconf` file in your template directory.
2. Navigate in the Structure view to find the `<Items>` element in the `<BusinessApplication>` element.
3. In the `<Items>` child element, modify the `Program` item to include a `dynamicProperties` attribute with a value of `icon`.

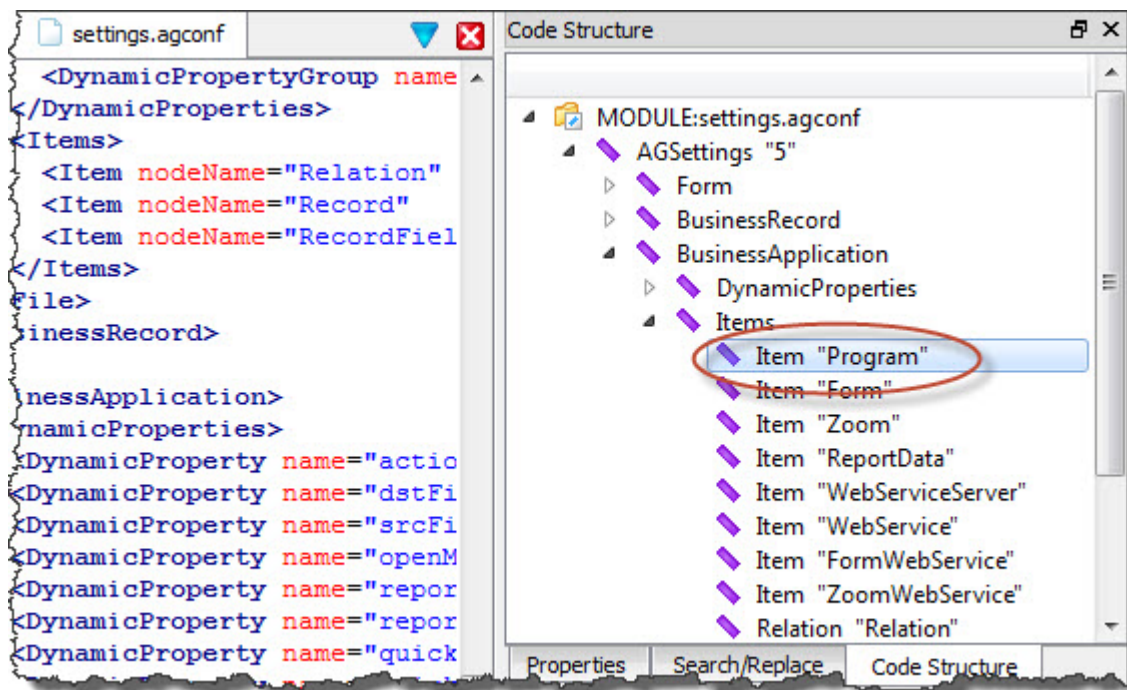


Figure 396: Structure view of settings.agconf.

```
<Item name="Program"
  label="New Program"
  extension=".4prg"
  icon="bullet_class"
  dynamicProperties="icon"/>
```

4. In the <DynamicProperties> section of the <BusinessApplication> element, add a new <DynamicProperty> element to define the new icon dynamic property.

```
<DynamicProperty name="icon"
  type="TEXT"
  label="Icon"
  description="Defines icon to be used in
  window title bar of running application."/>
```

5. Save your changes.
6. Select **Tools >> Specific setup >> Reload**.
7. Open the OfficeStore.4pw sample project from your My Genero Files directory.
8. Open the OfficeStoreAppFlow.4ba diagram and select the Account or Orders program entity on the diagram.
9. Note the new Icon property in the Properties view.

Modify tcl script to generate code

Modify the tcl script used to generate the code.

1. Open the main.tcl file in the tpl subdirectory of your template set directory.
2. Find the # Set XPath set of code. On the line following set ROOTFILELocator, add:

```
set icon [[getUniqueNode $ROOTNode {/AG/File/DynamicProperties/
DynamicProperty
  [@name='icon']}] @value]
```

- Find the instruction `[gen_MAIN $DBName $outFormRelNode]` and add the `$icon` parameter:

```
[gen_MAIN $DBName $outFormRelNode $icon]
```

- Find the instruction `proc gen_MAIN {DBName outFormRelNode}` { and add the `icon` parameter:

```
proc gen_MAIN {DBName outFormRelNode icon} {
```

- Find the instruction `CALL ui.Interface.loadStyles(\"dbapp\")` and on the following line add the call to set your image:

```
CALL ui.Interface.setImage(\"$icon\")
```

- Save the changes.
- Select **Tools >> Specific setup >> Reload**.
- In the `OfficestoreAppFlow.4ba`, select the **Account** program entity and set the value of the `Icon` property to a valid image name. See [Image directory structure](#) on page 952.
- Save the files.
- Rebuild and run the **Account** program and see the icon used in the window title of the program.

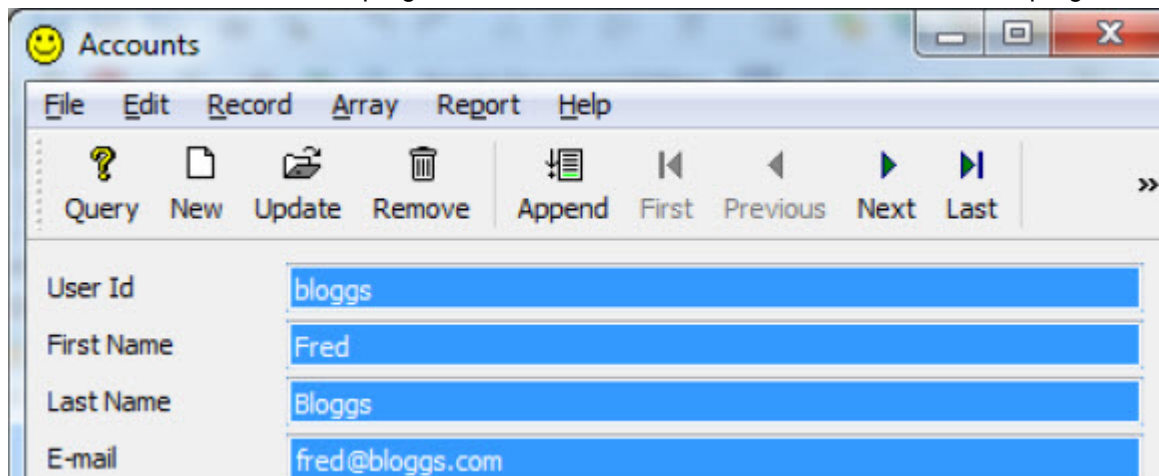


Figure 397: Custom icon in window title

Example 2: Adding a File >> New item

In this example, the **File >> New** dialog is modified to include a new, custom item.

Add a new file type definition to settings.agconf

Modify the `settings.agconf` file to include a new file type based on the `4dbx` database schema file type.

- Open the `settings.agconf` file in your template directory.
- Find the Database section. Notice that there is a File element defining the `4dbx` file type. Copy and paste this File element so that you have an additional File element in your Database element.
- Modify the extension attribute to use your own database extension, for example `4dbz`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<AGSettings version="5">
  <Form> </Form>
  <BusinessRecord> </BusinessRecord>
  <BusinessApplication> </BusinessApplication>
  <Database>
    <File extension="4dbx">
      <DynamicProperties>
```

```

        <DynamicProperty name="widget" type="ENUM" label="Widget"
            initialValue="Edit" description="Associated widget"
            editorInfo="contains:ButtonEdit|CheckBox|ComboBox|DateEdit|
Edit |
            FFImage|FFLabel|Field|Phantom|ProgressBar|RadioGroup|Slider|
            SpinEdit|TextEdit|TimeEdit"/>
        <DynamicProperty name="label" type="TEXT" label="Label"
            initialValue=""
            description="Associated label when generating form"/>
    </DynamicProperties>
    <Items>
        <Item nodeName="column" dynamicProperties="widget;label"/>
    </Items>
</File>
<File extension="4dbz">
    <DynamicProperties>
        <DynamicProperty name="widget" type="ENUM" label="Widget"
            initialValue="Edit" description="Associated widget"
            editorInfo="contains:ButtonEdit|CheckBox|ComboBox|DateEdit|
Edit |
            FFImage|FFLabel|Field|Phantom|ProgressBar|RadioGroup|Slider|
            SpinEdit|TextEdit|TimeEdit"/>
        <DynamicProperty name="label" type="TEXT" label="Label"
            initialValue="" description="Associated label when generating
form"/>
    </DynamicProperties>
    <Items>
        <Item nodeName="column" dynamicProperties="widget;label"/>
    </Items>
</File>
</Database>
</AGSettings>

```

4. Save your files.

Define an action in File >> New for the new item

Modify the creatables.conf file to define the new item in File >> New.

1. Open the creatables.conf file in your template directory.
2. Notice that the Category elements organize the creatables into groups. Find the Category element with a name attribute of Database. Modify the File element to represent the newly defined database file by changing the label, extension, and source attributes.

```

<Category index="30" label="Database" name="Database">
    <File index="30" name="DBXNewERD"
        action="DBOpen"
        label="DB Schema (.4dbz)"
        description="Create a new DB schema"
        icon="document_newSchema"
        extension="4dbz"
        source="creatables/dbapp.4dbz"
        isTemplate="true"/>
</Category>

```

3. Save the changes.
4. Since the source element was changed to creatables/dbapp.4dbz, create that file in the creatables subdirectory in your template directory. Copy and rename dbapp.4dbx to dbapp.4dbz.

Test new action

View the new addition to the New dialog.

1. Launch Genero Studio.

2. Select **Tools >> Specific setup >> Reload**.
3. Select **File >> New** and select your new database file type from the Database category.

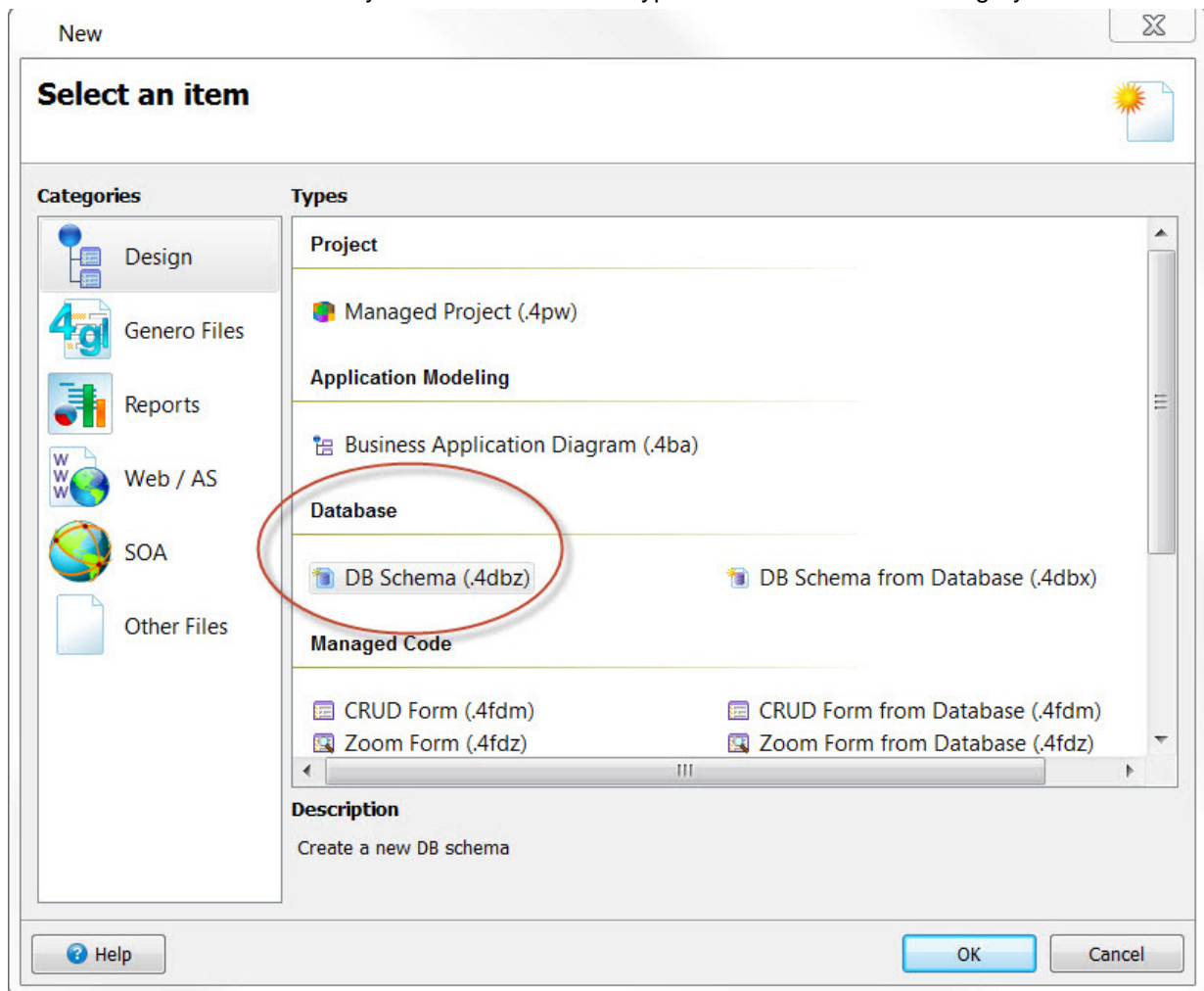


Figure 398: New item

Example 3: Adding an entity to the BA Diagram

In this example, the BA diagram is modified to include a new, custom item.

Add a new item to the BA diagram

Create a new Program entity called Start in the BA diagram.

1. Open the `settings.agconf` file in your template directory.
2. Find the `BusinessApplication` section. Add a new `DynamicProperty` and `Item`. In the `Item` element, define a new Program item with a name, extension, label, icon and `dynamicProperties` attribute values of your choice. The value of the `dynamicProperties` attribute corresponds with the name attribute in the `DynamicProperty` element.

```
<BusinessApplication>
  <DynamicProperties>
    <DynamicProperty
      name="myProp"
      type="TEXT"
      label="myProp" />
    </DynamicProperties>
  <Items>
```



```

    <Item
      name="Start"
      label="New Start item"
      extension="4srt"
      icon="bullet_class"
      dynamicProperties="myProp" />
  </Items>
</BusinessApplication>

```

3. Save the changes.
4. Select **Tools >> Specific setup >> Reload** to reload the modified template file.
5. Select the **Library** node in your project and right-click. Select **New >> Business Application Diagram (4ba)** to create a new 4ba.
6. Right-click on the BA diagram to see and select a new **Start** item. Save the 4ba file to your project directory (not the template directory).

The item cannot yet be implemented because the new file type (4srt) has not yet been defined.

Define the template file association

Add a new file type for the new program type.

1. Select **Tools >> Specific setup >> Edit File Associations**.
2. Define the file type and associated action. The `mimeType` and `extensions` attributes must match what you defined in your `settings.agconf` file.

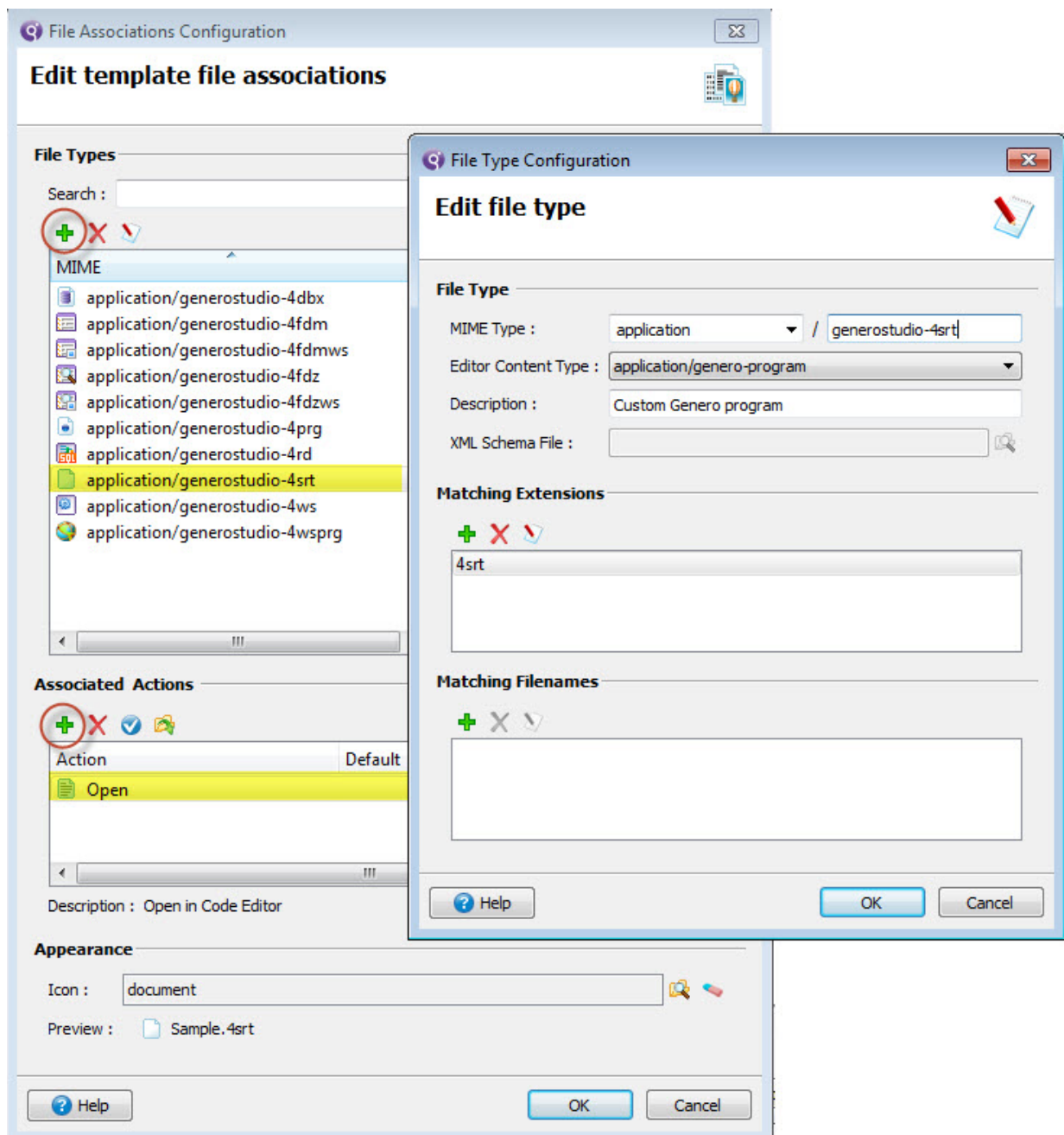


Figure 399: Adding a file type

3. Select OK to save the changes.

Add the new element to the program entity

Add the new element to the program entity.

1. Open the `creatables.conf` file in your template directory.
2. Find the `Managed Code` category and add a new `New` element to represent your new program entity.

```
<New
  index="900"
  name="CustomProgram"
  action="BANewProgram"
  label="Custom Program (.4srt)"
  actionLabel="Implement Custom Program"
```

```
icon="document_4prg"
description="Create Program"
extension="4srt" />
```

3. Save the changes.
4. Select **Tools >> Specific setup >> Reload** to reload the modified template file.
5. Close and re-open the new Business Application diagram (4ba) in the **Library** node of your project.
6. Right-click on the **Start** entity you added to your program (or add one). Notice the action to implement your program now exists in the menu. Select this action.

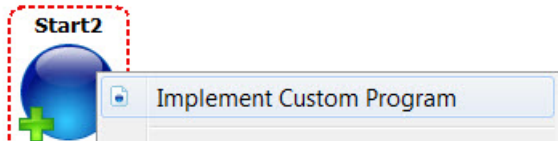


Figure 400: Implement new program entity

7. Save the 4srt file to the **Application** node in your project.

Define the build rules

Define the build rules for how the new file type is to be compiled.

1. Select **Tools >> Specific setup >> Edit Build Rules**.
2. Use the duplicate button to make a copy of the 4PRG build rule.
3. Edit the build rule to match your new program definition (4srt).

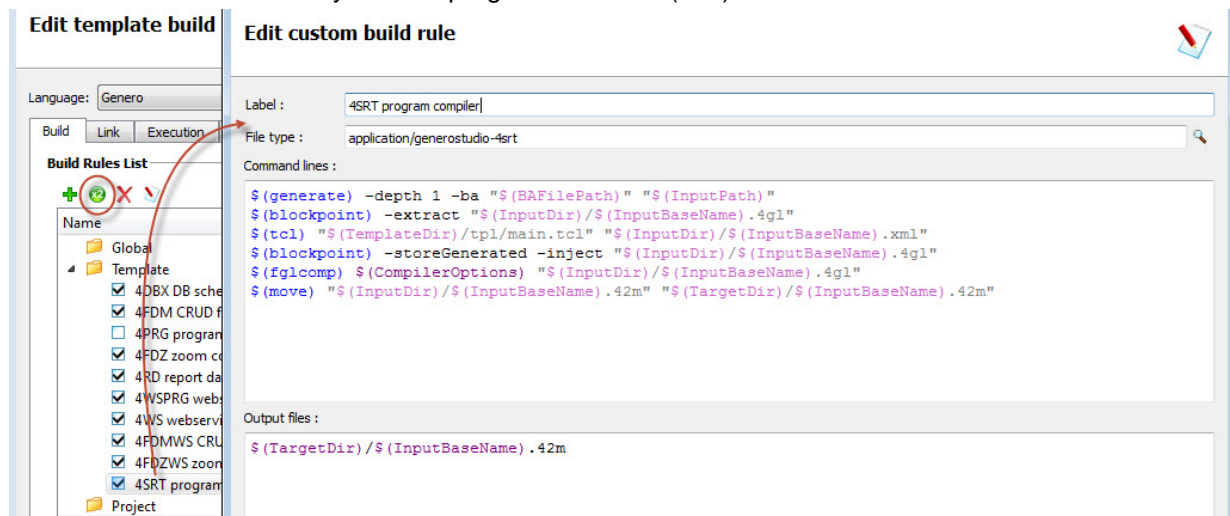


Figure 401: Creating a new build rule

4. Save.
5. If your modifications include an image other than the default, copy the image file to the folders in the *GSTDIR/images* directory. See [Image directory structure](#) on page 952.
6. You can now use your new 4SRT entity in your BA diagrams.

How code is generated

When you build an application from a Business Application diagram, the build rules define the various files that are input into the Code Generation Engine and the application code files that are output.

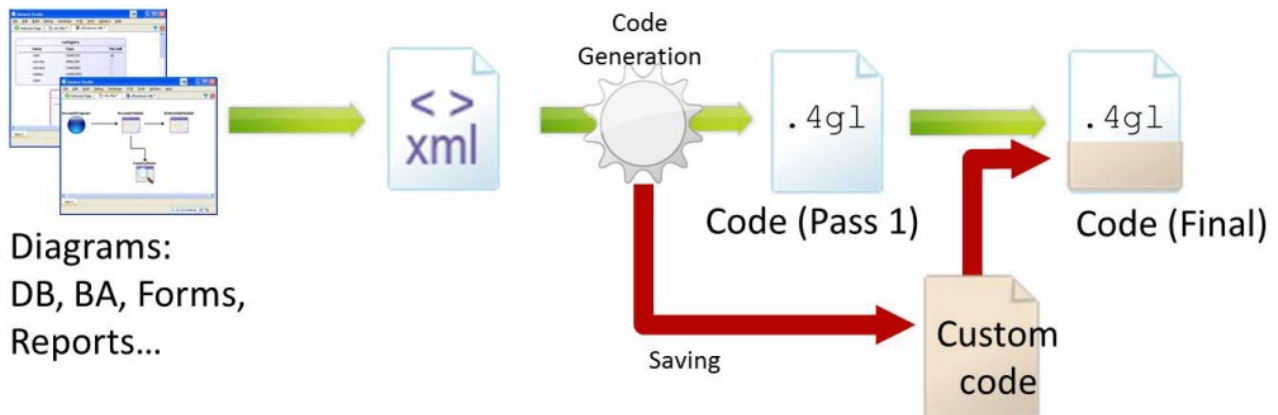


Figure 402: Code Generation flow

BAM Consolidates and Generates

Data Consolidation

The input from the BA diagram and related entities is gathered into a single XML file, which consolidates all the inputs into one package. This file is used when processing and generating the application code. This file could also be used to provide input to create the application models.

Code Generation

The XML file and a code template are used to generate the application code. The default Tcl template produces Genero 4gl files, but another tool could be used to generate the code (XSL translator for example). Custom code is preserved; if any custom code was created earlier, it is automatically restored in the newly generated application code.

Example

The build rules define the series of commands used to build and generate the code. In general, the Build rule for code generation:

- Saves custom code added by the user to the generated source files
- Generates the new source files without user code
- Restores the user code in the generated files
- Compiles the written and generated source files
- Links the compiled files

View default build rules selecting **Projects >> Edit Build Rules**. This example shows the build rules used to generate the code for a Program entity in the Business Application diagram.

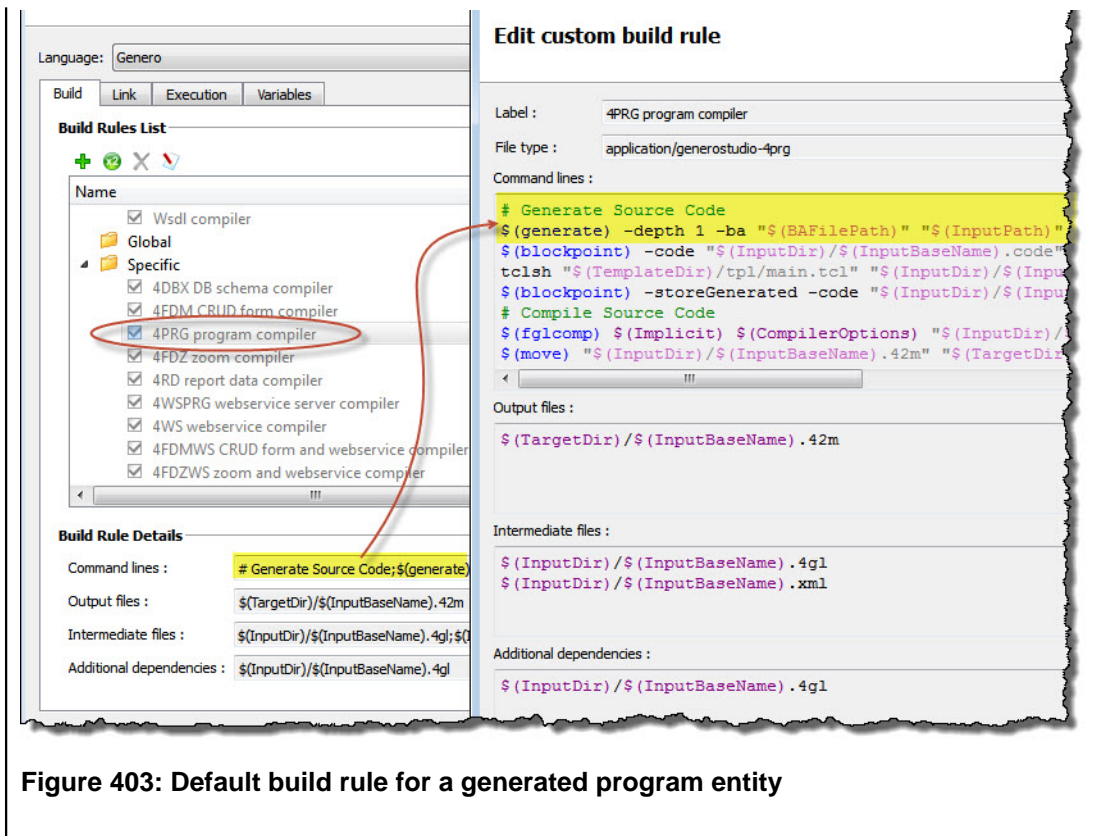


Figure 403: Default build rule for a generated program entity

Table 250: Default build rule example

Build rule command	Description
<code>\$(generate)</code>	The <code>\$(generate)</code> command creates an intermediary XML file from modeled entities.
<code>\$(blockpoint) -code</code>	BLOCK/POINT is extracted from previously generated and modified code.
<code>tclsh</code> on page 269	The <code>tclsh</code> executable generates the final file by using both a Tcl template file and the intermediary XML file created by the <code>\$(generate)</code> command.
<code>\$(blockpoint) -storeGenerated</code>	Extracted BLOCK/POINT code is put back into the generated code.
<code>\$(fglcomp)</code>	The <code>fglcomp</code> tool compiles BDL program sources files into a p-code version.
<code>\$(move)</code>	Moves the given file or directory to the given destination in a platform independent way.

Reviewing the Build

To better understand what is happening during the build of the program, turn on verbose mode using **Tools >> Preferences, Compiler and Runtime, Compilation Configuration**. Compile a diagram file, program, or application and view the results in the output.

The code generation template set

Genero Studio provides a standard template set of files that are used for code generation. The files are written in XML and Tcl.

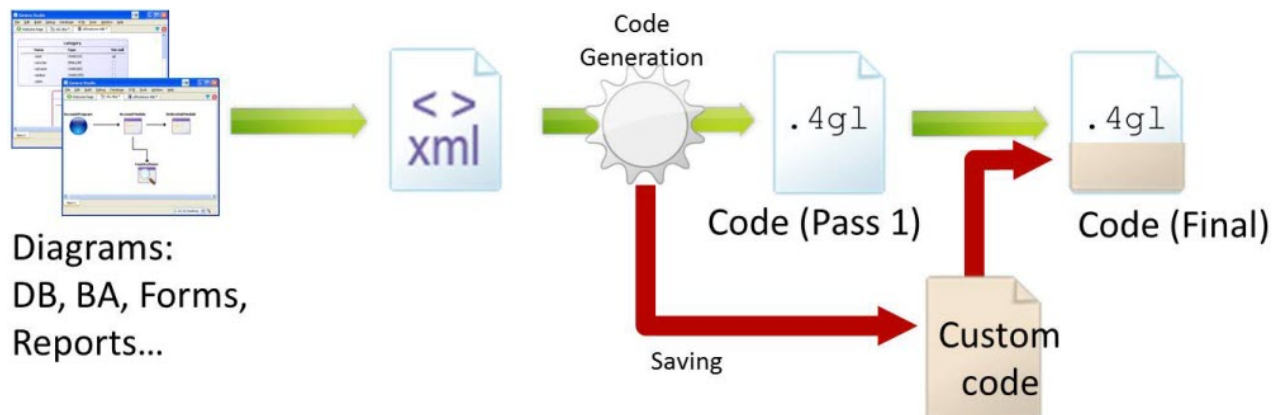


Figure 404: Code Generation flow

Default templates

Table 251: Application Generation Templates

Template	Description
dbapp3.1	This is the default and recommended template for 2.51.
dbapp3.0	This is the default and recommended template for 2.50.
dbapp2.0	This is the default and recommended template for 2.41.
dbapp1.0	The dbapp template in Genero Studio 2.40 was renamed in Genero Studio 2.41 to dbapp1.0. The dbapp template set is the same template set as dbapp1.0. This is the default and recommended template for 2.40. Using the <code>DIALOG</code> instruction, sub-dialog statements that display data, query a database, and edit single records and record lists can execute in parallel, allowing the application to handle different parts of a form simultaneously.
2.3x	This is a legacy template for compatibility with applications developed with Genero Studio 2.3x.

The `GSTSETUPDIR` environment variable specifies which template set is used.

Template directory contents

`GSTDIR/bin/src/ag/tpl/...` includes:

- [settings.agconf](#)
- [creatables.conf](#)
- [file-types.xml](#)

- build.rules
- *.tcl
- resource/**/*

Note: It is recommended that you not modify the original template files, but make a copy of the directory. Copy the entire content of the *GSTDIR/gst/bin/src/ag/tpl/dbapp* directory to a new directory. The new template directory could be a common one on the server or it could be part of the entire versioned project.

Interpreting settings.agconf

The settings.agconf file is an XML document with four sections: Form, Business Record, Business Application, and Database. It is used to manage the properties available to each of the diagrams used by the Business Application Modeler.

There are four main sections corresponding to the four diagrams that can be customized:

- [Form](#) on page 954
- [BusinessRecord](#) on page 963
- [BusinessApplication](#) on page 971
- [Database](#) on page 979

In the default settings.agconf, the Database section, for example, shows the information about the dynamicProperties defined for the Meta-schema file type (4dbx). The dynamicProperties found in the Meta-schema diagram for a 4dbx file are widget and label. When an item in the diagram, specified by nodeName, is selected, the properties appear as assigned by dynamicProperty. These properties are grouped according to the dynamicPropertyGroup element. These properties have a label as assigned by the label attribute and data type as specified by type attribute. The widget/wizard used is specified by elements such as editorInfo.

```
<Database>
  <File extension="4db">
    <DynamicProperties>
      <DynamicProperty name="widget" type="ENUM" label="Widget"
        initialValue="Edit" description="Associated widget"
        editorInfo="contains:ButtonEdit|CheckBox|ComboBox|DateEdit|Edit|
FFImage|
FFLabel|Field|ProgressBar|RadioGroup|Slider|SpinEdit|TextEdit|
TimeEdit"/>
      <DynamicProperty name="label" type="TEXT" label="Label"
        initialValue=""
        description="Associated label when generating form" />
      <DynamicPropertyGroup name="formfieldGroup" label="Formfield"
        description="Formfied properties" properties="label;widget" />
    </DynamicProperties>

    <Items>
      <Item nodeName="column" dynamicProperties="widget;label"/>
    </Items>
  </File>
</Database>
```

Name	Value
Object	
Name	orderid
Column	
Order	1
Data type	INTEGER
Not null	<input checked="" type="checkbox"/>
Default value	
Formfield	
Label	Order Id
Widget	Edit

Figure 405: Label and Widget properties in Meta-schema diagram (4db)

The WEB custom editor

Custom property editors are defined by the customer. The WEB custom editor is an HTML property editor.

Property info definition

The property info definition in settings.agconf:

- type="WEB"
- editorInfo:
 - htmlEditor : path to the html file relative to the template directory
 - isDebug : false : activates the webkit debug
 - title : title of the dialog (banner + window title)

Example

```
<DynamicProperty name="expression" type="WEB" label="expression"
  initialValue=""
  editorInfo="htmlEditor:expression.html;isDynamic:true;isDebug:false;title:Express
  editor"/>
```

HTML and JavaScript

property.value implicitly contains the property value when the editor is opened. The value present in property.value is taken as a new property value when the user has validated the dialog (and onEditorAccept() is called).

The function onDocumentSource(source) is called once the html document is loaded in the webkit. The source contains the source of the GeneroStudio current document (the document which contains the property).

The function onEditorAccept() is called when the user presses the OK button to accept the value.

The function onEditorCancel() is called when the user refuses the value (by pressing cancel or the x window button).

expression.html file

This is an example of an expression HTML file, referenced by the earlier DynamicProperty editor definition example.

```
<?xml version="1.0" encoding="utf-8" ?>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
  </head>
  <script language="JavaScript" type="text/javascript">
    // onDocumentSource event is called when the custom editor
is loaded
    // source : the complete document source in xml
function onDocumentSource(source)
{
    // find record fields list and populate the ui
var list = document.getElementById("fieldlist")
var parser = new DOMParser();
var xmlDoc = parser.parseFromString (source, "text/
xml");
var recordNodes = xmlDoc.getElementsByTagName
("RecordField");
for (var i = 0 ; i < recordNodes.length ; i++)
{
var recordNode = recordNodes[i];
var recordName = recordNode.getAttribute ("name");

// create one item for the recordNode field
var option=document.createElement("OPTION");
option.text = recordName;
option.value = recordName;
list.add(option);
}
// populate the input with the expression
var elem = document.getElementById("expression");
elem.value = property.value;
}

function pressed(character)
{
var elem = document.getElementById("expression");
elem.value += character;
property.value = elem.value;
}

function clearExpression()
{
var elem = document.getElementById("expression");
elem.value = "";
property.value = elem.value;
}

function selectField(fieldname)
{
var elem = document.getElementById("expression");
elem.value += fieldname;
property.value = elem.value;
}
```

```

    // onEditorAccept event is called before validating the
    custom editor
    function onEditorAccept()
    {
        alert("editor accepted !");
    }

    // onEditorCancel event is called before validating the
    custom editor
    function onEditorCancel()
    {
        alert("editor cancelled !");
    }
</script>
<body width="200" height="50">
<center>
<form name="calc" action="">
<table>
<b>
<table border=2 width=50 height=60 cellpadding=1 cellspacing=5>
<tr>
<td rowspan="4"><select size="8" id="fieldlist"
    onDbClick="selectField(this.value)" /></td>
<td><input name="btnSeven" type="Button" value=" 7 "
    onclick="pressed(7)"></td>
<td><input name="btnEight" type="Button" value=" 8 "
    onclick="pressed(8)"></td>
<td><input name="btnNine" type="Button" value=" 9 "
    onclick="pressed(9)"></td>
<td align="middle"><input name="btnPlus" type="Button" value="
+ " onclick="pressed('+)" /></td>
</tr>
<tr>
<td><input name="btnFour" type="Button" value=" 4 "
    onclick="pressed(4)"></td>
<td><input name="btnFive" type="Button" value=" 5 "
    onclick="pressed(5)"></td>
<td><input name="btnSix" type="Button" value=" 6 "
    onclick="pressed(6)"></td>
<td align="middle"><input name="btnMinus" type="Button" value="
- " onclick="pressed('-)" /></td>
</tr>
<tr>
<td><input name="btnOne" type="Button" value=" 1 "
    onclick="pressed('1)"></td>
<td><input name="btnTwo" type="Button" value=" 2 "
    onclick="pressed('2)"></td>
<td><input name="btnThree" type="Button" value=" 3 "
    onclick="pressed('3)"></td>
<td align="middle"><input name="btnMultiply" type="Button"
    value=" * " onclick="pressed('*)" /></td>
</tr>
<tr>
<td><input name="btnZero" type="Button" value=" 0 "
    onclick="pressed(0)"></td>
<td><input name="btnDecimal" type="Button" value=" . "
    onclick="pressed('.')"></td>
<td><input name="btnEquals" type="Button" value=" = "
    onclick="pressed('=)"></td>
<td align="middle"><input name="btnDivide" type="Button" value="
/ " onclick="pressed('/')"></td>
</tr>
<tr><td colspan="4"><input name="expression" type="Text"
    size="45" id="expression" /></td>

```

```

<td><input name="btnClear" type="Button" value=" C "
onclick="clearExpression()"/></td>
</tr>
</table>
</table>
</b>
</form>
</center>
</body>
</html>

```

The PROCESS custom editor

Custom property editors are defined by the customer. The PROCESS custom editor is a standalone executable property editor.

Property info definition

The property info definition in settings.agconf:

- type="PROCESS"
- editorInfo:
 - processEditor : path to the process editor executable relative to the template directory.

Example

```

<DynamicProperty name="process" type="PROCESS" label="process"
description="process"
editorInfo="processEditor:userPropertyEditor.exe;isDynamic:true"
dynamicContent="masterTable" />

```

Executable

The program's first argument is the property value when the editor opens.

If the program outputs a window id to the standard output, Genero Studio will ensure the corresponding window remains on top of Genero Studio (similar to a modal dialog). The syntax is a single line:

```
WindowId: <Window Identifier>
```

Once the property editing is done, the program should output the property value, on a single line as well:

```
Value: <property value>
```

Note: Non-printable characters must be escaped.

Interpreting creatables.conf

File >> New menu options can be defined in the `creatables.conf` file. Added items are called creatables.

Within the `creatables.conf` file there are Category elements. [Category](#) on page 988 elements correspond to a group of actions (creatables). Each [New](#), [File](#), [Directory](#), [Wizard](#) on page 989 element corresponds to a file type that can be created using the **File >> New** menu or context menu.

In the default `creatables.conf` file, one of the defined Creatables is the **Module Form from Database (4fdm)**, specifying its category (Design), subcategory (Managed Code), and type of Creadable (Wizard).

Example: Managed Form File Type

```
<Creatables version="1.0">
  <Category index="5" label="Design" name="MDA" icon="document_4ba" >
    <Category index="40" label="Managed Code" name="ManagedCode">
      <Wizard index="10"
        name="FDModuleForm"
        action="FDNew"
        label="Module Form from Database (4fdm)"
        icon="document_4fdm"
        description="Create an empty module in Form Designer"
        extension="4fdm" />
    </Category>
  </Category>
</Creatables>
```

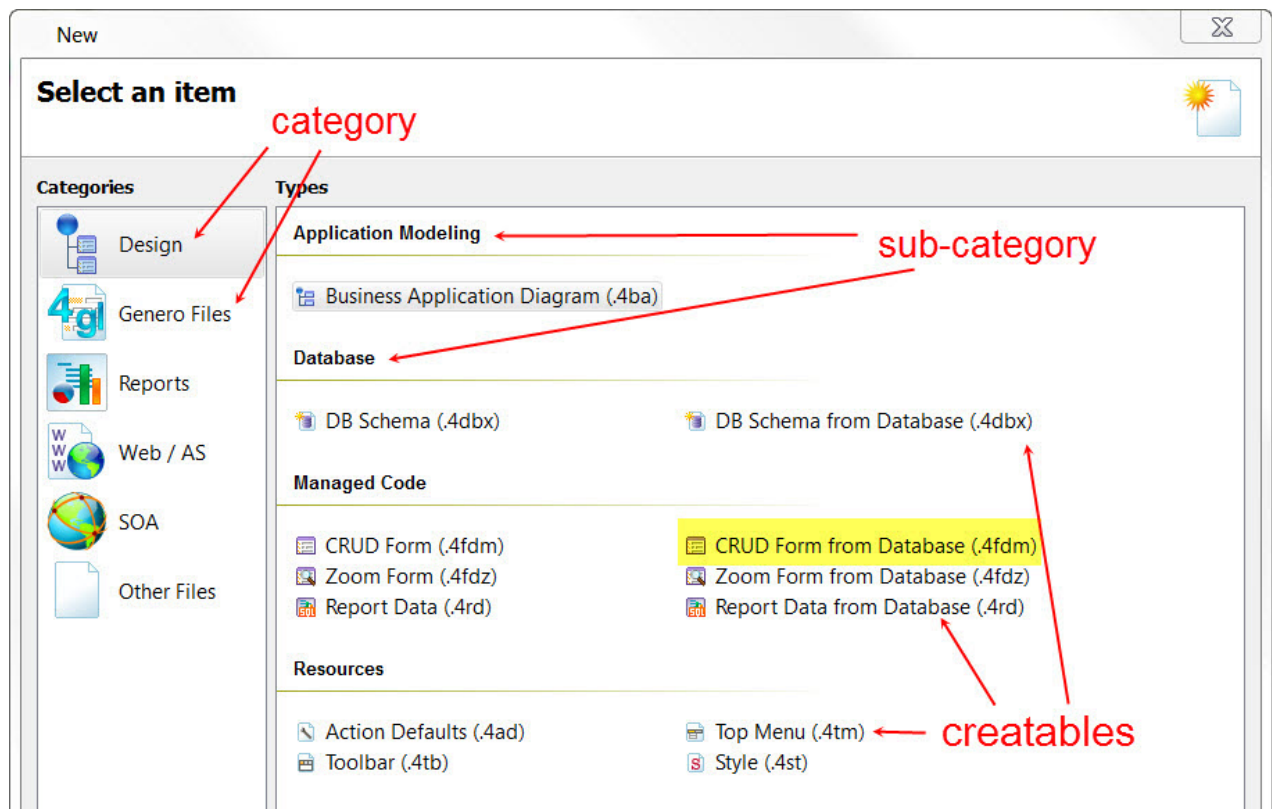


Figure 406: New Items: Creatables

Tcl basics and samples

A Tcl script is used to generate the final file using .tcl files and the intermediary .xml file generated during the build of the application. The tcl files can be modified to produce different 4gl code or to reference newly added properties.

Tcl (pronounced "tickle") is a scripting language commonly used for scripted applications and GUIs. You can use a scripting language of your choice or modify the included tcl files.

To determine which tcl file to modify turn on verbose mode for build, link, execution rules using **Tools >> Preferences, Compiler and Runtime**. Then, compile the diagram file that you want to modify. Find `tclsh` in the resulting output to see which `tcl` files are used.

Add POINT and BLOCK sections to template

You can add new POINT and BLOCK areas to a template.

1. To add a POINT or BLOCK section, determine where the POINT or BLOCK should be in the generated code and modify the (Tcl) file that corresponds to the generated file (4g1).

To better understand what is happening during the build of the program, turn on verbose mode using **Tools >> Preferences, Compiler and Runtime, Compilation Configuration**. You will be able to see what tcl files are being used and when. This can help you identify which tcl file you need to modify.

2. Add a new POINT or BLOCK section. Be sure to give the POINT or BLOCK a unique name that is not already used by another POINT or BLOCK.
3. Surround your added POINT or BLOCK area using the comment character of the targeted source code (such as "{" and "}" characters in Genero). Otherwise, the generated source code will not compile.
4. Open your project and rebuild.
5. To revert a change made in the source code, see [Revert a change to a POINT or BLOCK](#) on page 264.

Example: Using XSLT instead of Tcl

The code for your application can be generated using XSLT instead of Tcl.

This example demonstrates how to generate a file with AG using XSL Translator.

Generating a form

Instead of generating 4g1 as the template does, we'll generate a `per` file (Genero form definition file). This example uses a 4rd (report data) file and generate a `per` form file with formfields corresponding to the 4rd record. We'll add an additional property `frmLabel` on each record field, which will be displayed to the left of each `per` file formfield.

- The new property is added to the 4rd definition in `settings.agconf` (changes are in bold):

```
<File extension="4rd">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
label="foreignFields"
  initialValue="" dynamicContent="srcFieldsContent"
  editorInfo="isDynamic:true" />
    <DynamicProperty name="primaryFields" type="FIELDS"
label="primaryFields"
  initialValue="" dynamicContent="dstFieldsContent"
  editorInfo="isDynamic:true" />
    <DynamicProperty name="frmLabel" type="TEXT" label="Label FRM"
  initialValue="" />
  </DynamicProperties>
  <Items>
    <Item nodeName="Relation" srcProperty="foreignFields"
dstProperty="primaryFields"
dynamicProperties="primaryFields;foreignFields"/>
    <Item nodeName="RecordField" dynamicProperties="frmLabel" />
  </Items>
</File>
```

- The file `managedtextform.xsl` is created, to describe what is to be generated. This file should be saved to the template directory. The `GASDIR` environment variable should be defined in the GAS configuration and added to `PATH`.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
  <xsl:output method="text" />
  <xsl:template match="/">
--This is generated by the XSL
```

```

DATABASE <xsl:for-each select="//Module"><xsl:value-of
  select="@databaseName"/>
</xsl:for-each>

LAYOUT
GRID
{
  <xsl:for-each select="//Fields/Field">
    <xsl:value-of select="DynamicProperties/
DynamicProperty[@name='frmlabel']
  /@value"/>:
    [<xsl:value-of select="@column"/>]
  </xsl:for-each>
}
END -- LAYOUT

TABLES <xsl:for-each select="//Record"><xsl:value-of select="@table"/>
</xsl:for-each>

ATTRIBUTES
  <xsl:for-each select="//Fields/Field">
    <xsl:value-of select="@column"/> = <xsl:value-of select="@table"/
>.
    <xsl:value-of select="@column"/>;
  </xsl:for-each>

END -- ATTRIBUTES
</xsl:template>
</xsl:stylesheet>

```

- A new project is created and a new A Business Application diagram (4ba) is created and saved to the project.
- A new 4rd file is added to the project and a record created. The properties for **frmlabel** are added to the record fields.
- The 4rd Build Rule is updated to generate a `per` using `xslt`.

```

<BuildRule
  additionalDependencies=""
  commands="$(generate) -depth 0 -ba &quot;$(BAFilePath)&quot;
&quot;$(InputPath)&quot;;fglxslp -o
  $(InputDir)/$(InputBaseName)per $(GSTSETUPDIR)/managedTextForm.xml
  $(InputDir)/$(InputBaseName).xml;"
  description="4RD Report Data compiler"
  enabled="true"
  fileType="application/generostudio-4rd"
  id="1319189142826"
  intermediateFiles=""
  outputFiles="$(InputDir)/$(InputBaseName)per "/>

```

The same Build Rule displayed in a Genero Studio dialog:

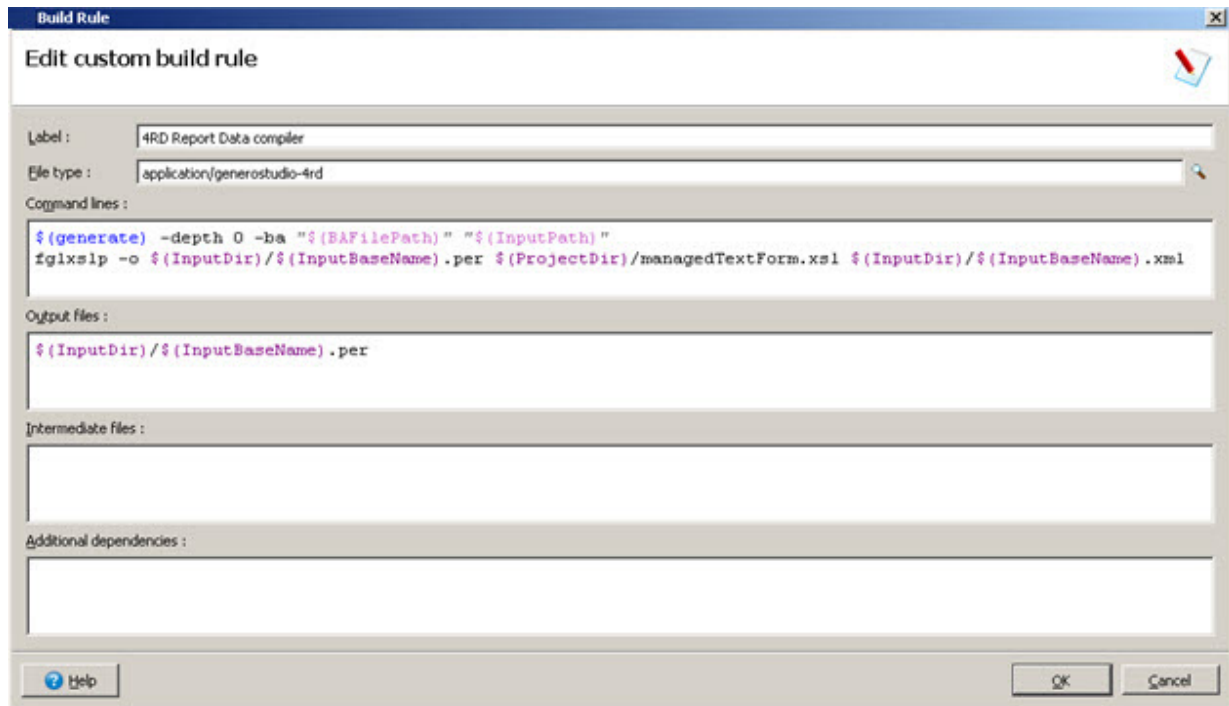


Figure 407: Build Rule

- The 4rd is compiled and a per is generated.

The generated form

When the per file is opened and previewed, it looks like this.

The image shows a Windows application window titled 'w1'. The window contains a form with the following fields and controls:

- user id :
- courriel :
- prenom :
- nom :
- status :
- adresse :
- adresse2 :
- ville :
- etat :
- code postal :
- pays :
- telephone :
- langue :
- categorie :
- liste :
- banniere :
- source :

On the right side of the form, there is a vertical panel titled 'Main' containing an 'Exit' button. At the bottom right corner of the window, there is a small logo that says 'INS'.

Figure 408: per form

Image directory structure

Genero Studio searches the *GSTDIR/images/* directories for images referenced in the templates and associated files.

A copy of the image, of increasing size, should be stored in the images directory in S1 through S5 size subdirectories.

- s1 : 16x16 pixels : structure view, Toolbar, file type icon (File Browser, Project Manager)
- s2 : 22x22 pixels
- s3® : 32x32 pixels
- s4 : 48x48 pixels : Business Application diagram
- s5 : 64x64 pixels

Depending on the DPI set on the system, other sizes can be chosen. Existing icons can be found in *GSTDIR/images/s**. These can be used in Business Application settings as well.

Template Reference

Reference topics for the default template set.

XML reference

Reference information for the XML template files.

- [settings.agconf elements](#) on page 953
- [creatables.conf elements](#) on page 987

settings.agconf elements

The settings.agconf file is an XML document with four sections: Form, Business Record, Business Application, and Database. It is used to manage the properties available to each of the diagrams used by the Business Application Modeler.

The file is located in the template directory *GSTDIR/gst/bin/src/ag/tpl/dbapp*.

The XML schema for settings.agconf is *GSTDIR/gst/conf/schema/agconf.xsd*.

- [AGSettings](#) on page 954
 - [Form](#) on page 954
 - [File](#) on page 955
 - [Messages](#) on page 956
 - [DynamicProperties](#)
 - [DynamicProperty](#) on page 957
 - [DynamicPropertyGroup](#) on page 961
 - [Items](#)
 - [Item \(Form / BusinessRecord / Database\)](#) on page 962
 - [BusinessRecord](#) on page 963
 - [File](#) on page 955
 - [Messages](#) on page 956
 - [DynamicProperties](#)
 - [DynamicProperty](#) on page 957
 - [Items](#)
 - [Item \(Form / BusinessRecord / Database\)](#) on page 962
 - [BusinessApplication](#) on page 971
 - [DynamicProperties](#)
 - [DynamicProperty](#) on page 957
 - [Items](#)
 - [Item \(BusinessApplication\)](#) on page 975
 - [Relation](#) on page 977
 - [Constraints](#)
 - [Constraint](#) on page 978
 - [Database](#) on page 979
 - [File](#) on page 955
 - [DynamicProperties](#)
 - [DynamicProperty](#) on page 957

- Items
 - [Item \(Form / BusinessRecord / Database\)](#) on page 962

AGSettings

The `AGSettings` element is the root element of the `settings.agconf` file.

The `AGSettings` element takes an optional attribute `version`.

Syntax

```
<AGSettings version="5">
  <Form></Form>
  <BusinessRecord></BusinessRecord>
  <BusinessApplication></BusinessApplication>
  <Database></Database>
</AGSettings>
```

Child elements

The `AGSettings` element may contain the following child elements:

1. Zero or one [Form](#) element.
2. Zero or one [BusinessRecord](#) element.
3. Zero or one [BusinessApplication](#) element.
4. Zero or one [Database](#) element.

Example

See `GSTDIR/bin/src/ag/tpl/dbapp/settings.agconf`

Form

The Form section in the `settings.agconf` file manages the properties available in Form Designer.

The `Form` element defines the format of a form type.

Syntax

```
<Form>
  <File attributes>
    <DynamicProperties> dynamic property list </DynamicProperties>
    <Items> item list </Items>
  </File>
</Form>
```

Child elements

The `Form` element may contain the following child elements:

1. One or more [File](#) elements.

Example

```
<Form>
  <File extension="4fdm" isManaged="true">
    <DynamicProperties>
      <DynamicProperty
        name="T_QUERY"
        type="BOOLEAN"
        label="canSearch"
        initialValue="1"
        description="Allow search using Query By Example"/>
    </DynamicProperties>
  </File>
</Form>
```

```

    <DynamicPropertyGroup
      name="user"
      label="Functionality"
      description="User properties group"

      properties="T_QUERY " />
  </DynamicProperties>
  <Items>
    <Item
      nodeName="Record"
      dynamicProperties="T_QUERY" />
  </Items>
</File>
</Form>

```

File

The `File` element defines the file type. Add one `File` element for each new file type.

Syntax

```

<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicPropertyGroup/>
  </DynamicProperties>
  <Items>
    <Item/>
  </Items>
</File>

```

Attributes

Table 252: File element attributes

Attribute	Options
extension	File extension.
isManaged	Form element only. True or False (Default is False). Set to True if the type corresponds to managed form (to generate application code using records features like queries). If managed, the records in the form are enabled by default, otherwise they cannot be used for Application Modeling.

Child elements

The `File` element may contain the following child elements:

1. One or more [DynamicProperty](#) on page 957 and [DynamicPropertyGroup](#) on page 961 elements, within one `DynamicProperties` element.
2. One or more [Item](#) elements, within one `Items` element.

Example - File elements representing a file type.

```

<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent" />
  </DynamicProperties>
</File>

```

```

<DynamicProperty name="primaryFields" type="FIELDS"
  label="primaryFields" initialValue=""
  editorInfo="isDynamic:true" dynamicContent="dstFieldsContent"/>
<DynamicPropertyGroup name="relations"
  label="Relation"
  description="Relation properties group"
  properties="foreignFields;primaryFields"/>
<DynamicProperty name="canSearch" type="BOOLEAN"
  label="canSearch" initialValue="true"
  description="Allow search using Query By Example"/>
<DynamicProperty name="canAdd" type="BOOLEAN"
  label="canAdd" initialValue="true"
  description="Allow adding items"/>
<DynamicProperty name="canModify" type="BOOLEAN"
  label="canModify" initialValue="true"
  description="Allow modifying existing items"/>
<DynamicProperty name="canDelete" type="BOOLEAN"
  label="canDelete" initialValue="true"
  description="Allow deleting items"/>
<DynamicPropertyGroup name="functionality"
  label="Functionality"
  description="Functionality properties group"
  properties="canAdd;canModify;canDelete;canSearch"/>
</DynamicProperties>
<Items>
  <Item nodeName="Relation" srcProperty="foreignFields"
    dstProperty="primaryFields"
    dynamicProperties="primaryFields;foreignFields"/>
  <Item nodeName="Record"
    dynamicProperties="canAdd;canModify;canDelete;canSearch"/>
</Items>
</File>

```

Messages

The `Messages` element is used to identify, for a specific item, error messages to disable during code generation and/or to reduce from an error to a warning.

The `Messages` element has attributes.

Attributes

Table 253: Messages element attributes

Attribute	Options
<code>warningAsError</code>	A semicolon separated list of error identifiers. If an error identifier is added to an item, the message severity is increased from warning to error for this specific error for this specific item.
<code>ignoreWarning</code>	A semicolon separated list of error identifiers. If an error identifier is added to an item, the specified error is no longer generated for this specific item. Disables the message if its severity is warning.

Example

```

<Database>
  <File extension="4db">
    <Message warningAsError="GS-11000" ignoreWarning
      ="GS-11360;GS-11361">
    <DynamicProperties>

```

DynamicProperty

The `DynamicProperty` element is a child element of `DynamicProperties`.

`DynamicProperties` can have one or more `DynamicProperty` children.

The `DynamicProperty` element has attributes. Valid attributes depend on the parent element of `Form`, `BusinessRecord`, `BusinessApplication`, or `Database`.

Syntax

```
<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicProperty/>
    <DynamicProperty/>
  </DynamicProperties>
</File>
```

Attributes

Table 254: DynamicProperty element attributes

Attribute	Options
name	Property identifier, name that appears in the xml. Must be unique among all the dynamic properties. The property gets associated with the Item / Relation through name.
description	Textual description of the property. Displayed in property view tooltip.
type	Property type. Defines how the property editor behaves. <ul style="list-style-type: none"> BOOLEAN The editor is a checkbox. Valid values are true, false. MULTIPLEBOOLEAN The editor is the multiple checkbox dialog, editorInfo can have contains(list of checkable values), title, description and icon ENUM The editor is combobox, values are in editorInfo contains: or dynamic INTEGER The editor is spinbox , range is in editorInfo: range TEXT The editor is for text. FILE The editor is the browse dialog. FIELDS The editor is the field selection dialog. The fields are dynamic. BRQUERY The editor is the record query dialog. This property type is reserved, do not use. FDTEXT This property type is reserved, do not use. FDSTYLE This property type is reserved, do not use.

Attribute	Options
	<p>FDINCLUDE This property type is reserved, do not use.</p> <p>FDCOLORCONDITION This property type is reserved, do not use.</p> <p>MULTIPLELINES This editor is for text with a dialog for multiple line editing.</p> <p>MULTIPLEPATH This editor is the multiple path dialog. The <code>hasBrowseButton</code> specifies if the editor should have browse button. <code>editPathMode</code> can be file or directory. <code>selectionMode</code> can be single or multiple.</p> <p>A <i>custom property</i> is a property which uses a property editor defined by the customer. We provide only 2 types of custom editors:</p> <p>WEB The editor is a input field with a browse button which opens a dialog box displaying a html document. See The WEB custom editor on page 944.</p> <p>PROCESS The editor is a input field with a browse button which launches an external process. See The PROCESS custom editor on page 947.</p>
<code>initialValue</code>	Default value used when no user value is set. This is the value set when the restore button is used. The default value is not saved in the file.
<code>dynamicContent (Form)</code>	<p>Name of the dynamic content source if the property is dynamic. The available dynamic contents in Form Designer are:</p> <p>databaseName List of available databases in the project and preferences.</p> <p>sqlTableName List of available database tables in the current database.</p> <p>colName List of available columns available with the <code>sqlTableName</code> table in the current database</p> <p>aggregateColName List of available columns available with the <code>aggregateTableName</code> table in the current database.</p> <p>displayColName List of available columns available with the <code>displayLikeTableName</code> table in the current database.</p>

Attribute	Options
	<p>validateColName List of available columns available with the <i>validateLikeTableName</i> table in the current database.</p> <p>fieldType List of available field type (only formonly if there is no database).</p> <p>hidden Adds <code>USER</code> to true, false if the selection contains only table columns.</p> <p>posX, posY, gridwith, gridHeight Returns the geometry limits depending on parent geometry.</p> <p>rowCount, colCount, stepX, stepY Corresponding min and max (for matrices).</p> <p>totalRows Min and max rows for table elements.</p> <p>expandedColumns Deprecated.</p> <p>masterTable List of record implicit tables.</p> <p>componentType List of available web components.</p> <p>lookup List of lookups in the parent record.</p> <p>srcFieldsContent List of available record fields in relation source.</p> <p>dstFieldsContent List of available record fields in relation destination.</p> <p>precision Available precision for current <i>qual1</i> qualifier.</p> <p>scale Available scale for current <i>qual2</i> qualifier.</p> <p>aggregatePrecision Available aggregate precision for current <i>aggregateQual1</i> qualifier.</p> <p>children List of available children names.</p> <p>sibling List of available sibling names.</p>
<p><code>dynamicContent</code> (BusinessApplication)</p>	<p>Name of the dynamic content source if the property is dynamic.</p> <p>The available dynamic contents in BusinessApplication are:</p> <p>srcFieldsContent List of available fields in the relation source or the current item if not a relation.</p> <p>dstFieldsContent List of available fields in the destination item or the current item if not a relation.</p>

Attribute	Options
	<p>actionContent List of available actions in the source item or the current item if not a relation.</p> <p>type List of available types for the current element.</p>
editorInfo	<p>Semi-colon separated list of <code>attribute:value</code> pair containing the property editor information.</p> <p>For example:</p> <pre>editorInfo="contains:ButtonEdit CheckBox ComboBox DateEdit Edit FFImage FFLabel Field ProgressBar RadioGroup Slider SpinEdit TextEdit TimeEdit"</pre> <p>editor Use another editor than the default one associated to the property type.</p> <p>alwaysUpdate If true, the model is modified immediately on each user action.</p> <p>contains List of values for ENUM type if not dynamic.</p> <p>editMode If true, the property editor is editable (combobox, BRQuery).</p> <p>filters For FILE type, set the browse dialog extension filter.</p> <p>isDynamic If true, the values (contains property) is computed.</p> <p>range Range of values for the spinbox editor (similar to contains), two values separated with ' '. </p> <p>icon Contains the banner icon for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p> <p>title Contains the window title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p> <p>description Contains the banner title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p>
label	Text displayed in the Properties view.
readOnly	Sets property to read only. Options are <code>true</code> or <code>false</code> .
isHidden	Sets visibility of the property. Options are <code>true</code> or <code>false</code> .

Example

```
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
    <DynamicProperty name="primaryFields" type="FIELDS"
      label="primaryFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="dstFieldsContent"/>
    <DynamicPropertyGroup name="relations"
      label="Relation"
      description="Relation properties group"
      properties="foreignFields;primaryFields"/>
    <DynamicProperty name="canSearch" type="BOOLEAN"
      label="canSearch" initialValue="true"
      description="Allow search using Query By Example"/>
    <DynamicProperty name="canAdd" type="BOOLEAN"
      label="canAdd" initialValue="true"
      description="Allow adding items"/>
    <DynamicProperty name="canModify" type="BOOLEAN"
      label="canModify" initialValue="true"
      description="Allow modifying existing items"/>
    <DynamicProperty name="canDelete" type="BOOLEAN"
      label="canDelete" initialValue="true"
      description="Allow deleting items"/>
    <DynamicPropertyGroup name="functionality"
      label="Functionality"
      description="Functionality properties group"
      properties="canAdd;canModify;canDelete;canSearch"/>
  </DynamicProperties>
  <Items>
    <Item nodeName="Relation" srcProperty="foreignFields"
      dstProperty="primaryFields"
      dynamicProperties="primaryFields;foreignFields"/>
    <Item nodeName="Record"
      dynamicProperties="canAdd;canModify;canDelete;canSearch"/>
  </Items>
</File>
```

DynamicPropertyGroup

The `DynamicPropertyGroup` element is a child element of `DynamicProperties` and is used to organize a group of properties together.

The `DynamicPropertyGroup` element has attributes.

Syntax

```
<DynamicProperties>
  </DynamicPropertyGroup>
</DynamicProperties>
```

Table 255: DynamicPropertyGroup element attributes

Attribute	Options
name	Property identifier, name that appears in the xml.

Attribute	Options
description	Textual description of the property. Displayed in Property view tooltip.
label	Text displayed in the Properties view.
properties	List of dyanmicProperty names to display in the group, separated by semi-colons.

Example

```
<DynamicPropertyGroup name="functionality" label="Functionality"
    description="Functionality properties group"
    properties="canAdd;canModify;canDelete;canSearch;canDisplay;canEmpty" />
```

Item (Form / BusinessRecord / Database)

The `Item` element adds new dynamic properties to some existing node(s) in the Form or Business Record or Meta-schema, depending on `nodeName`.

The `Item` element has attributes.

Table 256: Item element attributes

Attribute	Options
nodeName	Corresponds to the node type name, such as "Relation".
srcProperty	Name of the property used as the source fields list, if the item is a Relation on page 977. Defines the source field while creating the query in application generation. The value of this property is the name of the dynamic property which holds source field. This dynamic property must be associated with relation through the <code>dynamicProperty</code> attribute.
dstProperty	Name of the property used as the destination fields list, if the item is a Relation on page 977. Similar to <code>srcProperty</code> it defines the destination field.
dynamicProperties	List of dynamic properties that apply to the item node.

Syntax

```
<BusinessRecord>
  <File extension="4rd">
    <DynamicProperties>
      <DynamicProperty name="foreignFields" type="FIELDS"
        label="foreignFields"
        initialValue="" dynamicContent="srcFieldsContent"
        editorInfo="isDynamic:true" />
      <DynamicProperty name="primaryFields" type="FIELDS"
        label="primaryFields"
        initialValue="" dynamicContent="dstFieldsContent"
        editorInfo="isDynamic:true" />
    </DynamicProperties>
    <Items>
      <Item
        nodeName="Relation"
        srcProperty="foreignFields"
        dstProperty="primaryFields"
        dynamicProperties="primaryFields;foreignFields" />
    </Items>
  </File>
</BusinessRecord>
```

```

</Items>
</File>
</BusinessRecord>

```

BusinessRecord

Similar to `Form`, the `BusinessRecord` element defines the settings for a Business Record entity, which describes the database columns for a report. The only difference is that the `isManaged` attribute does not exist (all Business Record files are managed).

In this default format, the only item that differs from the Record associated with a form is a Relation, and the Dynamic Properties apply to that item.

Syntax

```

<BusinessRecord>
  <File attributes>
    <DynamicProperties> dynamic property list </DynamicProperties>
    <Items> item list </Items>
  </File>
</BusinessRecord>

```

Child elements

The `BusinessRecord` element may contain the following child elements:

1. One or more `File` elements.

Example

```

<BusinessRecord>
  <File extension="4rd">
    <DynamicProperties>
      <DynamicProperty name="foreignFields" type="FIELDS"
label="foreignFields"
initialValue="" dynamicContent="srcFieldsContent"
editorInfo="isDynamic:true" />
      <DynamicProperty name="primaryFields" type="FIELDS"
label="primaryFields"
initialValue="" dynamicContent="dstFieldsContent"
editorInfo="isDynamic:true" />
    </DynamicProperties>
    <Items>
      <Item nodeName="Relation" srcProperty="foreignFields"
dstProperty="primaryFields"
dynamicProperties="primaryFields;foreignFields" />
    </Items>
  </File>
</BusinessRecord>

```

File

The `File` element defines the file type. Add one `File` element for each new file type.

Syntax

```

<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicPropertyGroup/>
  </DynamicProperties>
<Items>

```

```

</Item/>
</Items>
</File>

```

Attributes

Table 257: File element attributes

Attribute	Options
extension	File extension.
isManaged	Form element only. True or False (Default is False). Set to True if the type corresponds to managed form (to generate application code using records features like queries). If managed, the records in the form are enabled by default, otherwise they cannot be used for Application Modeling.

Child elements

The File element may contain the following child elements:

1. One or more [DynamicProperty](#) on page 957 and [DynamicPropertyGroup](#) on page 961 elements, within one [DynamicProperties](#) element.
2. One or more [Item](#) elements, within one [Items](#) element.

Example - File elements representing a file type.

```

<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
    <DynamicProperty name="primaryFields" type="FIELDS"
      label="primaryFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="dstFieldsContent"/>
    <DynamicPropertyGroup name="relations"
      label="Relation"
      description="Relation properties group"
      properties="foreignFields;primaryFields"/>
    <DynamicProperty name="canSearch" type="BOOLEAN"
      label="canSearch" initialValue="true"
      description="Allow search using Query By Example"/>
    <DynamicProperty name="canAdd" type="BOOLEAN"
      label="canAdd" initialValue="true"
      description="Allow adding items"/>
    <DynamicProperty name="canModify" type="BOOLEAN"
      label="canModify" initialValue="true"
      description="Allow modifying existing items"/>
    <DynamicProperty name="canDelete" type="BOOLEAN"
      label="canDelete" initialValue="true"
      description="Allow deleting items"/>
    <DynamicPropertyGroup name="functionality"
      label="Functionality"
      description="Functionality properties group"
      properties="canAdd;canModify;canDelete;canSearch"/>
  </DynamicProperties>
  <Items>
    <Item nodeName="Relation" srcProperty="foreignFields"
      dstProperty="primaryFields"

```

```

        dynamicProperties="primaryFields;foreignFields"/>
        <Item nodeName="Record"
            dynamicProperties="canAdd;canModify;canDelete;canSearch"/>
    </Items>
</File>

```

Messages

The `Messages` element is used to identify, for a specific item, error messages to disable during code generation and/or to reduce from an error to a warning.

The `Messages` element has attributes.

Attributes

Table 258: Messages element attributes

Attribute	Options
<code>warningAsError</code>	A semicolon separated list of error identifiers. If an error identifier is added to an item, the message severity is increased from warning to error for this specific error for this specific item.
<code>ignoreWarning</code>	A semicolon separated list of error identifiers. If an error identifier is added to an item, the specified error is no longer generated for this specific item. Disables the message if its severity is warning.

Example

```

<Database>
  <File extension="4db">
    <Message warningAsError="GS-11000" ignoreWarning
      ="GS-11360;GS-11361">
      <DynamicProperties>

```

DynamicProperty

The `DynamicProperty` element is a child element of `DynamicProperties`.

`DynamicProperties` can have one or more `DynamicProperty` children.

The `DynamicProperty` element has attributes. Valid attributes depend on the parent element of `Form`, `BusinessRecord`, `BusinessApplication`, or `Database`.

Syntax

```

<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicProperty/>
    <DynamicProperty/>
  </DynamicProperties>
</File>

```

Attributes

Table 259: DynamicProperty element attributes

Attribute	Options
name	Property identifier, name that appears in the xml. Must be unique among all the dynamic properties. The property gets associated with the Item / Relation through name.
description	Textual description of the property. Displayed in property view tooltip.
type	<p>Property type. Defines how the property editor behaves.</p> <p>BOOLEAN The editor is a checkbox. Valid values are true, false.</p> <p>MULTIPLEBOOLEAN The editor is the multiple checkbox dialog, editorInfo can have contains(list of checkable values), title, description and icon</p> <p>ENUM The editor is combobox, values are in editorInfo contains: or dynamic</p> <p>INTEGER The editor is spinbox , range is in editorInfo: range</p> <p>TEXT The editor is for text.</p> <p>FILE The editor is the browse dialog.</p> <p>FIELDS The editor is the field selection dialog. The fields are dynamic.</p> <p>BRQUERY The editor is the record query dialog. This property type is reserved, do not use.</p> <p>FDTEXT This property type is reserved, do not use.</p> <p>FDSTYLE This property type is reserved, do not use.</p> <p>FDINCLUDE This property type is reserved, do not use.</p> <p>FDCOLORCONDITION This property type is reserved, do not use.</p> <p>MULTIPLELINES This editor is for text with a dialog for multiple line editing.</p> <p>MULTIPLEPATH This editor is the multiple path dialog. The <code>hasBrowseButton</code> specifies if the editor should have browse button. <code>editPathMode</code> can be file or directory. <code>selectionMode</code> can be single or multiple.</p>

Attribute	Options
	<p>totalRows Min and max rows for table elements.</p> <p>expandedColumns Deprecated.</p> <p>masterTable List of record implicit tables.</p> <p>componentType List of available web components.</p> <p>lookup List of lookups in the parent record.</p> <p>srcFieldsContent List of available record fields in relation source.</p> <p>dstFieldsContent List of available record fields in relation destination.</p> <p>precision Available precision for current <i>qual1</i> qualifier.</p> <p>scale Available scale for current <i>qual2</i> qualifier.</p> <p>aggregatePrecision Available aggregate precision for current <i>aggregateQual1</i> qualifier.</p> <p>children List of available children names.</p> <p>sibling List of available sibling names.</p>
dynamicContent (BusinessApplication)	<p>Name of the dynamic content source if the property is dynamic.</p> <p>The available dynamic contents in BusinessApplication are:</p> <p>srcFieldsContent List of available fields in the relation source or the current item if not a relation.</p> <p>dstFieldsContent List of available fields in the destination item or the current item if not a relation.</p> <p>actionContent List of available actions in the source item or the current item if not a relation.</p> <p>type List of available types for the current element.</p>
editorInfo	<p>Semi-colon separated list of <code>attribute:value</code> pair containing the property editor information.</p> <p>For example:</p> <pre>editorInfo="contains:ButtonEdit CheckBox ComboBox DateEdit Edit FFImage FFLabel Field ProgressBar RadioGroup Slider SpinEdit TextEdit TimeEdit"</pre>

Attribute	Options
	<p>editor Use another editor than the default one associated to the property type.</p> <p>alwaysUpdate If true, the model is modified immediately on each user action.</p> <p>contains List of values for ENUM type if not dynamic.</p> <p>editMode If true, the property editor is editable (combobox, BRQuery).</p> <p>filters For FILE type, set the browse dialog extension filter.</p> <p>isDynamic If true, the values (contains property) is computed.</p> <p>range Range of values for the spinbox editor (similar to contains), two values separated with ' '. </p> <p>icon Contains the banner icon for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p> <p>title Contains the window title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p> <p>description Contains the banner title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p>
label	Text displayed in the Properties view.
readOnly	Sets property to read only. Options are true or false.
isHidden	Sets visibility of the property. Options are true or false.

Example

```
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
    <DynamicProperty name="primaryFields" type="FIELDS"
      label="primaryFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="dstFieldsContent"/>
    <DynamicPropertyGroup name="relations"
      label="Relation"
      description="Relation properties group"
      properties="foreignFields;primaryFields"/>
    <DynamicProperty name="canSearch" type="BOOLEAN"
      label="canSearch" initialValue="true"
      description="Allow search using Query By Example"/>
  </DynamicProperties>
</File>
```

```

<DynamicProperty name="canAdd" type="BOOLEAN"
  label="canAdd" initialValue="true"
  description="Allow adding items"/>
<DynamicProperty name="canModify" type="BOOLEAN"
  label="canModify" initialValue="true"
  description="Allow modifying existing items"/>
<DynamicProperty name="canDelete" type="BOOLEAN"
  label="canDelete" initialValue="true"
  description="Allow deleting items"/>
<DynamicPropertyGroup name="functionality"
  label="Functionality"
  description="Functionality properties group"
  properties="canAdd;canModify;canDelete;canSearch"/>
</DynamicProperties>
<Items>
  <Item nodeName="Relation" srcProperty="foreignFields"
    dstProperty="primaryFields"
    dynamicProperties="primaryFields;foreignFields"/>
  <Item nodeName="Record"
    dynamicProperties="canAdd;canModify;canDelete;canSearch"/>
</Items>
</File>

```

Item (Form / BusinessRecord / Database)

The `Item` element adds new dynamic properties to some existing node(s) in the Form or Business Record or Meta-schema, depending on `nodeName`.

The `Item` element has attributes.

Table 260: Item element attributes

Attribute	Options
<code>nodeName</code>	Corresponds to the node type name, such as "Relation".
<code>srcProperty</code>	Name of the property used as the source fields list, if the item is a Relation on page 977. Defines the source field while creating the query in application generation. The value of this property is the name of the dynamic property which holds source field. This dynamic property must be associated with relation through the <code>dynamicProperty</code> attribute.
<code>dstProperty</code>	Name of the property used as the destination fields list, if the item is a Relation on page 977. Similar to <code>srcProperty</code> it defines the destination field.
<code>dynamicProperties</code>	List of dynamic properties that apply to the item node.

Syntax

```

<BusinessRecord>
  <File extension="4rd">
    <DynamicProperties>
      <DynamicProperty name="foreignFields" type="FIELDS"
        label="foreignFields"
        initialValue="" dynamicContent="srcFieldsContent"
        editorInfo="isDynamic:true" />
      <DynamicProperty name="primaryFields" type="FIELDS"
        label="primaryFields"
        initialValue="" dynamicContent="dstFieldsContent"
        editorInfo="isDynamic:true" />
    </DynamicProperties>
    <Items>
      <Item

```

```

        nodeName="Relation"
        srcProperty="foreignFields"
        dstProperty="primaryFields"
        dynamicProperties="primaryFields;foreignFields" />
    </Items>
</File>
</BusinessRecord>

```

BusinessApplication

The `BusinessApplication` element defines the format of all the entities (items) that a BA diagram can contain, and their dynamic properties.

Unlike the Form and Business Record formats, the Item elements in this file do not modify the behavior of an existing item, they create new item definitions with new names, inheriting from a default item. The items are Program, Form, Zoom, ReportData, WebService and WebServiceServer. This format is similar to the BusinessRecord, except there is no File element.

DynamicProperty

The `DynamicProperty` element is a child element of `DynamicProperties`.

`DynamicProperties` can have one or more `DynamicProperty` children.

The `DynamicProperty` element has attributes. Valid attributes depend on the parent element of Form, BusinessRecord, BusinessApplication, Or Database.

Syntax

```

<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicProperty/>
    <DynamicProperty/>
  </DynamicProperties>
</File>

```

Attributes

Table 261: DynamicProperty element attributes

Attribute	Options
name	Property identifier, name that appears in the xml. Must be unique among all the dynamic properties. The property gets associated with the Item / Relation through name.
description	Textual description of the property. Displayed in property view tooltip.
type	Property type. Defines how the property editor behaves. BOOLEAN The editor is a checkbox. Valid values are true, false. MULTIPLEBOOLEAN The editor is the multiple checkbox dialog, editorInfo can have contains(list of checkable values), title, description and icon ENUM The editor is combobox, values are in editorInfo contains: or dynamic

Attribute	Options
	<p>INTEGER The editor is spinbox , range is in editorInfo: range</p> <p>TEXT The editor is for text.</p> <p>FILE The editor is the browse dialog.</p> <p>FIELDS The editor is the field selection dialog. The fields are dynamic.</p> <p>BRQUERY The editor is the record query dialog. This property type is reserved, do not use.</p> <p>FDTEXT This property type is reserved, do not use.</p> <p>FDSTYLE This property type is reserved, do not use.</p> <p>FDINCLUDE This property type is reserved, do not use.</p> <p>FDCOLORCONDITION This property type is reserved, do not use.</p> <p>MULTIPLELINES This editor is for text with a dialog for multiple line editing.</p> <p>MULTIPLEPATH This editor is the multiple path dialog. The <code>hasBrowseButton</code> specifies if the editor should have browse button. <code>editPathMode</code> can be file or directory. <code>selectionMode</code> can be single or multiple.</p> <p><i>A custom property is a property which uses a property editor defined by the customer. We provide only 2 types of custom editors:</i></p> <p>WEB The editor is a input field with a browse button which opens a dialog box displaying a html document. See The WEB custom editor on page 944.</p> <p>PROCESS The editor is a input field with a browse button which launches an external process. See The PROCESS custom editor on page 947.</p>
<code>initialValue</code>	Default value used when no user value is set. This is the value set when the restore button is used. The default value is not saved in the file.
<code>dynamicContent (Form)</code>	<p>Name of the dynamic content source if the property is dynamic. The available dynamic contents in Form Designer are:</p> <p>databaseName List of available databases in the project and preferences.</p>

Attribute	Options
sqlTabName	List of available database tables in the current database.
colName	List of available columns available with the <i>sqlTabName</i> table in the current database
aggregateColName	List of available columns available with the <i>aggregateTableName</i> table in the current database.
displayColName	List of available columns available with the <i>displayLikeTableName</i> table in the current database.
validateColName	List of available columns available with the <i>validateLikeTableName</i> table in the current database.
fieldType	List of available field type (only formonly if there is no database).
hidden	Adds <code>USER</code> to true, false if the selection contains only table columns.
posX, posY, gridwith, gridHeight	Returns the geometry limits depending on parent geometry.
rowCount, colCount, stepX, stepY	Corresponding min and max (for matrices).
totalRows	Min and max rows for table elements.
expandedColumns	Deprecated.
masterTable	List of record implicit tables.
componentType	List of available web components.
lookup	List of lookups in the parent record.
srcFieldsContent	List of available record fields in relation source.
dstFieldsContent	List of available record fields in relation destination.
precision	Available precision for current <i>qual1</i> qualifier.
scale	Available scale for current <i>qual2</i> qualifier.
aggregatePrecision	Available aggregate precision for current <i>aggregateQual1</i> qualifier.

Attribute	Options
	<p>children List of available children names.</p> <p>sibling List of available sibling names.</p>
dynamicContent (BusinessApplication)	<p>Name of the dynamic content source if the property is dynamic.</p> <p>The available dynamic contents in BusinessApplication are:</p> <p>srcFieldsContent List of available fields in the relation source or the current item if not a relation.</p> <p>dstFieldsContent List of available fields in the destination item or the current item if not a relation.</p> <p>actionContent List of available actions in the source item or the current item if not a relation.</p> <p>type List of available types for the current element.</p>
editorInfo	<p>Semi-colon separated list of <code>attribute:value</code> pair containing the property editor information.</p> <p>For example:</p> <pre data-bbox="662 1003 1302 1115">editorInfo="contains:ButtonEdit CheckBox ComboBox DateEdit Edit FFImage FFLabel Field ProgressBar RadioGroup Slider SpinEdit TextEdit TimeEdit"</pre> <p>editor Use another editor than the default one associated to the property type.</p> <p>alwaysUpdate If true, the model is modified immediately on each user action.</p> <p>contains List of values for ENUM type if not dynamic.</p> <p>editMode If true, the property editor is editable (combobox, BRQuery).</p> <p>filters For FILE type, set the browse dialog extension filter.</p> <p>isDynamic If true, the values (contains property) is computed.</p> <p>range Range of values for the spinbox editor (similar to contains), two values separated with ' '.</p> <p>icon Contains the banner icon for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p>

Attribute	Options
	<p>title Contains the window title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p> <p>description Contains the banner title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p>
label	Text displayed in the Properties view.
readOnly	Sets property to read only. Options are true or false.
isHidden	Sets visibility of the property. Options are true or false.

Example

```
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=" "
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
    <DynamicProperty name="primaryFields" type="FIELDS"
      label="primaryFields" initialValue=" "
      editorInfo="isDynamic:true" dynamicContent="dstFieldsContent"/>
    <DynamicPropertyGroup name="relations"
      label="Relation"
      description="Relation properties group"
      properties="foreignFields;primaryFields"/>
    <DynamicProperty name="canSearch" type="BOOLEAN"
      label="canSearch" initialValue="true"
      description="Allow search using Query By Example"/>
    <DynamicProperty name="canAdd" type="BOOLEAN"
      label="canAdd" initialValue="true"
      description="Allow adding items"/>
    <DynamicProperty name="canModify" type="BOOLEAN"
      label="canModify" initialValue="true"
      description="Allow modifying existing items"/>
    <DynamicProperty name="canDelete" type="BOOLEAN"
      label="canDelete" initialValue="true"
      description="Allow deleting items"/>
    <DynamicPropertyGroup name="functionality"
      label="Functionality"
      description="Functionality properties group"
      properties="canAdd;canModify;canDelete;canSearch"/>
  </DynamicProperties>
  <Items>
    <Item nodeName="Relation" srcProperty="foreignFields"
      dstProperty="primaryFields"
      dynamicProperties="primaryFields;foreignFields"/>
    <Item nodeName="Record"
      dynamicProperties="canAdd;canModify;canDelete;canSearch"/>
  </Items>
</File>
```

Items

The Items element defines the items on the BA diagram and the relations between them.

Item (BusinessApplication)

Unlike Form and BusinessRecord, the `Item` element of BusinessApplication does not modify the behavior of an existing item, it defines a new item, with a new name, inheriting from a default item.

The `Item` element takes attributes.

Table 262: Item element attributes

Attribute	Options
name	The new item type name.
label	Label for item as it will appear in the BA diagram New >> context menu.
extension	File extension.
icon	The icon used for the item in the Business Application diagram. The file must be present in the image directory of the Application Generator template directory. See Image directory structure on page 952.
dynamicProperties	List of dynamic properties dynamic properties that apply to the item node.
depth	For example, in a BA diagram specify the number of relations to traverse while generating the model. Outgoing relations are traversed. The model always contains the incoming and outgoing relation for the current item, but if the depth limit is reached, the target item definition is not generated. The depth is an integer starting from 0, or <code>unlimited</code> to traverse the complete application.

Syntax

```
<BusinessApplication>
  <DynamicProperties> dynamic property list </DynamicProperties>
  <Items> item list </Items>
  <Relation relation properties />
</BusinessApplication>
```

BasicItem

The `BasicItem` element defines a new item, with a new name. It does not modify the behavior of an existing item.

Table 263: BasicItem element attributes

Attribute	Options
name	The new basic item type name.
label	Label for item as it will appear in the BA diagram New >> context menu.
icon	The icon used for the item in the Business Application diagram. The file must be present in the image directory of the Application Generator template directory. See Image directory structure on page 952.
depth	For example, in a BA diagram specify the number of relations to traverse while generating the model. Outgoing relations are traversed. The model always contains the incoming and outgoing relation for the current item, but if the depth limit is reached, the target item definition is not

Attribute	Options
	generated. The depth is an integer starting from 0, or unlimited to traverse the complete application.

Syntax

```
<BusinessApplication>
  <DynamicProperties> dynamic property list </DynamicProperties>
  <Items>
    <Item> </Item>
    <BasicItem </BasicItem>
    <Relation> </Relation>
  </Items>
</BusinessApplication>
```

Relation

The Relation element defines a BA diagram relation.

Table 264: Relation element attributes

Attribute	Options
name	Defines the type of relation. There can be multiple relations of the type defined by name in BA diagram. The name must be unique in both item and relation definition.
dynamicProperties	List of dynamic properties that apply to the item node.

Syntax

```
<BusinessApplication>
  <DynamicProperties> dynamic property list </DynamicProperties>
  <Items>
    <Item> </Item>
    <BasicItem> </BasicItem>
    <Relation> </Relation>
  </Items>
</BusinessApplication>
```

Example

```
<Items>
  <Item name="Program" label="Program" extension="4prg"
  icon="bullet_class"/>
  <Item name="Form" label="CRUD Form" extension="4fdm"
  icon="document_4fdm" dynamicProperties=""/> <!--dynamic properties here -->
  <Item name="Zoom" label="Zoom Form" extension="4fdz"
  icon="document_4fdz" dynamicProperties=""/> <!--dynamic properties here -->
  <BasicItem name="Gallery" label="Gallery" icon="ba_choosePhoto"/>
  <BasicItem name="Phone" label="Phone" icon="ba_callPhone"/>
  <BasicItem name="Mail" label="Mail" icon="ba_composeMail"/>
  <BasicItem name="SMS" label="SMS" icon="ba_composeSms"/>
  <BasicItem name="Contact" label="Contact" icon="ba_chooseContact"/>
  <BasicItem name="Maps" label="Maps" icon="ba_mapsTo"/>
  <Relation name="Relation" dynamicProperties=""/> <!--dynamic
  properties here -->
  <Relation name="PhoneRelation"
  dynamicProperties="action/basicItemPhoneSrcField"/>
```

```

    <Relation name="MailRelation"    dynamicProperties="" /> <!--dynamic
properties here -->
    <Relation name="SMSRelation"
dynamicProperties="action;basicItemSMSToSrcField;basicItemSMSContentSrcField" /
>
    <Relation name="MapsRelation"
dynamicProperties="action;basicItemMapsSrcField" />
    <Relation name="PhotoRelation"
dynamicProperties="action;basicItemImagePathSrcField" />
    <Relation name="GalleryRelation"
dynamicProperties="action;basicItemImagePathSrcField" />
    <Relation name="ContactRelation"
dynamicProperties="action;basicItemVcardSrcField" />
</Items>

```

Constraint

Constraints are rules which apply to the relations between items on the BA diagram. If a constraint is not respected on an item, the corresponding error is displayed in the document errors view. During code generation, broken constraints generate errors and stop the generate process with an error status.

Constraints are used in the BA diagram to regulate what relations are valid between entities. For example, with the dbapp template set, if you create a Program and a Form entity on the BA diagram, a valid relation can be set between the Program and Form.

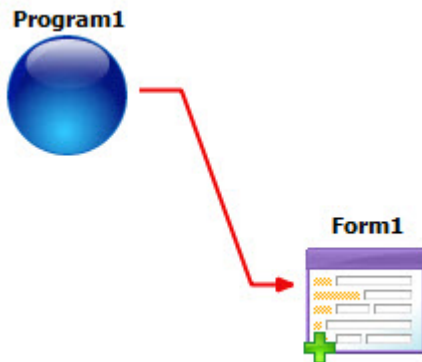


Figure 409: Valid relation between Program and CRUD Form

However, if you try to set a relation from a Form entity to a Program, you will see the constraint icon



indicating that this relation is not valid.

Each Constraint element manages a constraint on a specific type of relation: Relation, ReportRelation, or WebServiceRelation.

Table 265: Constraint element attributes

Attribute	Value	Description
name	Any string	Name of the constraint. Use to display the name in case of error.
reference	Semi-colon separated list such as Relation;ReportRelation;WebServiceRelation	Possible relation types, or * for any.
description	Any string	Textual description of the error.
source	Semi-colon separated list of source items from which the relation begins, such as ReportData;WebService;Program;Form.	Possible source item types, or * for any.

Attribute	Value	Description
destination	Semi-colon separated list of destination items to which the relation ends, such as Form;Zoom;FormWebService;ZoomWebService.	Possible destination item types, or * for any.
minSource	int >= 0 or '*' for unbounded	The minimum number of relation sources required for this relation type. Minimum source relation cardinality.
maxSource	int >= 0 or *' for unbounded	The maximum number of relation sources allowed for this relation type. Maximum source relation cardinality.
minDestination	int >= 0 or '*' for unbounded	Minimum destination relation cardinality.
maxDestination	int >= 0 or '*' for unbounded	The maximum number of relation destinations allowed for this relation type. Maximum destination relation cardinality. If set to 0, this means a relation that meets source and destination constraints cannot be used.

Example

```
<Constraint
  name="Constraint_010"
  reference="WebServiceRelation"
  description="Define at least one outgoing relationship from
WebServiceServer"
  source="WebServiceServer"
  destination="WebService;FormWebService;ZoomWebService"
  minSource="1"/>
```

In this example, Constraint_10 specifies that a WebServiceServer source entity must have at least one WebServiceRelation to a destination of WebService, FormWebservice or ZoomWebService.

Database

This Database section defines the default format for the items that have been added to the Genero Studio meta-schema file.

The widget and label properties are added to the column item in this section.

```
<Database>
  <File extension="4db">
    <DynamicProperties>
      <DynamicProperty name="widget" type="ENUM" label="Widget"
        initialValue="Edit" description="Associated widget"
        editorInfo="contains:ButtonEdit|CheckBox|ComboBox|DateEdit|Edit|
FFImage|
FFLabel|Field|ProgressBar|RadioGroup|Slider|SpinEdit|TextEdit|
TimeEdit"/>
      <DynamicProperty name="label" type="TEXT" label="Label"
        initialValue=""
        description="Associated label when generating form" />
    <DynamicPropertyGroup name="formfieldGroup" label="Formfield"
```

```

        description="Formfied properties" properties="label;widget" />
    </DynamicProperties>

    <Items>
        <Item nodeName="column" dynamicProperties="widget;label" />
    </Items>
</File>
</Database>

```

File

The `File` element defines the file type. Add one `File` element for each new file type.

Syntax

```

<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicPropertyGroup/>
  </DynamicProperties>
  <Items>
    <Item/>
  </Items>
</File>

```

Attributes

Table 266: File element attributes

Attribute	Options
extension	File extension.
isManaged	Form element only. True or False (Default is False). Set to True if the type corresponds to managed form (to generate application code using records features like queries). If managed, the records in the form are enabled by default, otherwise they cannot be used for Application Modeling.

Child elements

The `File` element may contain the following child elements:

1. One or more [DynamicProperty](#) on page 957 and [DynamicPropertyGroup](#) on page 961 elements, within one `DynamicProperties` element.
2. One or more [Item](#) elements, within one `Items` element.

Example - File elements representing a file type.

```

<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent" />
    <DynamicProperty name="primaryFields" type="FIELDS"
      label="primaryFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="dstFieldsContent" />
    <DynamicPropertyGroup name="relations"
      label="Relation"
      description="Relation properties group"

```

```

    properties="foreignFields;primaryFields"/>
    <DynamicProperty name="canSearch" type="BOOLEAN"
      label="canSearch" initialValue="true"
      description="Allow search using Query By Example"/>
    <DynamicProperty name="canAdd" type="BOOLEAN"
      label="canAdd" initialValue="true"
      description="Allow adding items"/>
    <DynamicProperty name="canModify" type="BOOLEAN"
      label="canModify" initialValue="true"
      description="Allow modifying existing items"/>
    <DynamicProperty name="canDelete" type="BOOLEAN"
      label="canDelete" initialValue="true"
      description="Allow deleting items"/>
    <DynamicPropertyGroup name="functionality"
      label="Functionality"
      description="Functionality properties group"
      properties="canAdd;canModify;canDelete;canSearch"/>
  </DynamicProperties>
  <Items>
    <Item nodeName="Relation" srcProperty="foreignFields"
      dstProperty="primaryFields"
      dynamicProperties="primaryFields;foreignFields"/>
    <Item nodeName="Record"
      dynamicProperties="canAdd;canModify;canDelete;canSearch"/>
  </Items>
</File>

```

Messages

The `Messages` element is used to identify, for a specific item, error messages to disable during code generation and/or to reduce from an error to a warning.

The `Messages` element has attributes.

Attributes

Table 267: Messages element attributes

Attribute	Options
<code>warningAsError</code>	A semicolon separated list of error identifiers. If an error identifier is added to an item, the message severity is increased from warning to error for this specific error for this specific item.
<code>ignoreWarning</code>	A semicolon separated list of error identifiers. If an error identifier is added to an item, the specified error is no longer generated for this specific item. Disables the message if its severity is warning.

Example

```

<Database>
  <File extension="4db">
    <Message warningAsError="GS-11000" ignoreWarning
      ="GS-11360;GS-11361">
    </DynamicProperties>
  </File>
</Database>

```

DynamicProperty

The `DynamicProperty` element is a child element of `DynamicProperties`.

`DynamicProperties` can have one or more `DynamicProperty` children.

The `DynamicProperty` element has attributes. Valid attributes depend on the parent element of `Form`, `BusinessRecord`, `BusinessApplication`, or `Database`.

Syntax

```
<File>
  <DynamicProperties>
    <DynamicProperty/>
    <DynamicProperty/>
    <DynamicProperty/>
  </DynamicProperties>
</File>
```

Attributes

Table 268: DynamicProperty element attributes

Attribute	Options
name	Property identifier, name that appears in the xml. Must be unique among all the dynamic properties. The property gets associated with the Item / Relation through name.
description	Textual description of the property. Displayed in property view tooltip.
type	Property type. Defines how the property editor behaves. <ul style="list-style-type: none"> BOOLEAN The editor is a checkbox. Valid values are true, false. MULTIPLEBOOLEAN The editor is the multiple checkbox dialog, editorInfo can have contains(list of checkable values), title, description and icon ENUM The editor is combobox, values are in editorInfo contains: or dynamic INTEGER The editor is spinbox , range is in editorInfo: range TEXT The editor is for text. FILE The editor is the browse dialog. FIELDS The editor is the field selection dialog. The fields are dynamic. BRQUERY The editor is the record query dialog. This property type is reserved, do not use. FDTEXT This property type is reserved, do not use. FDSTYLE This property type is reserved, do not use. FDINCLUDE This property type is reserved, do not use.

Attribute	Options
	<p>FDCOLORCONDITION This property type is reserved, do not use.</p> <p>MULTIPLELINES This editor is for text with a dialog for multiple line editing.</p> <p>MULTIPLEPATH This editor is the multiple path dialog. The <code>hasBrowseButton</code> specifies if the editor should have browse button. <code>editPathMode</code> can be file or directory. <code>selectionMode</code> can be single or multiple.</p> <p>A <i>custom property</i> is a property which uses a property editor defined by the customer. We provide only 2 types of custom editors:</p> <p>WEB The editor is a input field with a browse button which opens a dialog box displaying a html document. See The WEB custom editor on page 944.</p> <p>PROCESS The editor is a input field with a browse button which launches an external process. See The PROCESS custom editor on page 947.</p>
initialValue	Default value used when no user value is set. This is the value set when the restore button is used. The default value is not saved in the file.
dynamicContent (Form)	<p>Name of the dynamic content source if the property is dynamic. The available dynamic contents in Form Designer are:</p> <p>databaseName List of available databases in the project and preferences.</p> <p>sqlTabName List of available database tables in the current database.</p> <p>colName List of available columns available with the <code>sqlTabName</code> table in the current database</p> <p>aggregateColName List of available columns available with the <code>aggregateTableName</code> table in the current database.</p> <p>displayColName List of available columns available with the <code>displayLikeTableName</code> table in the current database.</p> <p>validateColName List of available columns available with the</p>

Attribute	Options
	<p><i>validateLikeTableName</i> table in the current database.</p> <p>fieldType List of available field type (only formonly if there is no database).</p> <p>hidden Adds <code>USER</code> to true, false if the selection contains only table columns.</p> <p>posX, posY, gridwith, gridHeight Returns the geometry limits depending on parent geometry.</p> <p>rowCount, colCount, stepX, stepY Corresponding min and max (for matrices).</p> <p>totalRows Min and max rows for table elements.</p> <p>expandedColumns Deprecated.</p> <p>masterTable List of record implicit tables.</p> <p>componentType List of available web components.</p> <p>lookup List of lookups in the parent record.</p> <p>srcFieldsContent List of available record fields in relation source.</p> <p>dstFieldsContent List of available record fields in relation destination.</p> <p>precision Available precision for current <i>qual1</i> qualifier.</p> <p>scale Available scale for current <i>qual2</i> qualifier.</p> <p>aggregatePrecision Available aggregate precision for current <i>aggregateQual1</i> qualifier.</p> <p>children List of available children names.</p> <p>sibling List of available sibling names.</p>
<p><code>dynamicContent</code> (BusinessApplication)</p>	<p>Name of the dynamic content source if the property is dynamic.</p> <p>The available dynamic contents in BusinessApplication are:</p> <p>srcFieldsContent List of available fields in the relation source or the current item if not a relation.</p> <p>dstFieldsContent List of available fields in the destination item or the current item if not a relation.</p> <p>actionContent List of available actions in the source item or the current item if not a relation.</p>

Attribute	Options
	<p>type List of available types for the current element.</p>
editorInfo	<p>Semi-colon separated list of <code>attribute:value</code> pair containing the property editor information.</p> <p>For example:</p> <pre>editorInfo="contains:ButtonEdit CheckBox ComboBox DateEdit Edit FFImage FFLabel Field ProgressBar RadioGroup Slider SpinEdit TextEdit TimeEdit"</pre> <p>editor Use another editor than the default one associated to the property type.</p> <p>alwaysUpdate If true, the model is modified immediately on each user action.</p> <p>contains List of values for ENUM type if not dynamic.</p> <p>editMode If true, the property editor is editable (combobox, BRQuery).</p> <p>filters For FILE type, set the browse dialog extension filter.</p> <p>isDynamic If true, the values (contains property) is computed.</p> <p>range Range of values for the spinbox editor (similar to contains), two values separated with ' '. </p> <p>icon Contains the banner icon for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p> <p>title Contains the window title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p> <p>description Contains the banner title for property editor opening a dialog box (text, checkboxlist, MULTIPLELINES)</p>
label	Text displayed in the Properties view.
readOnly	Sets property to read only. Options are true or false.
isHidden	Sets visibility of the property. Options are true or false.

Example

```
<File extension="4fdm" isManaged="true">
  <DynamicProperties>
    <DynamicProperty name="foreignFields" type="FIELDS"
      label="foreignFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="srcFieldsContent"/>
    <DynamicProperty name="primaryFields" type="FIELDS"
      label="primaryFields" initialValue=""
      editorInfo="isDynamic:true" dynamicContent="dstFieldsContent"/>
    <DynamicPropertyGroup name="relations"
      label="Relation"
      description="Relation properties group"
      properties="foreignFields;primaryFields"/>
    <DynamicProperty name="canSearch" type="BOOLEAN"
      label="canSearch" initialValue="true"
      description="Allow search using Query By Example"/>
    <DynamicProperty name="canAdd" type="BOOLEAN"
      label="canAdd" initialValue="true"
      description="Allow adding items"/>
    <DynamicProperty name="canModify" type="BOOLEAN"
      label="canModify" initialValue="true"
      description="Allow modifying existing items"/>
    <DynamicProperty name="canDelete" type="BOOLEAN"
      label="canDelete" initialValue="true"
      description="Allow deleting items"/>
    <DynamicPropertyGroup name="functionality"
      label="Functionality"
      description="Functionality properties group"
      properties="canAdd;canModify;canDelete;canSearch"/>
  </DynamicProperties>
  <Items>
    <Item nodeName="Relation" srcProperty="foreignFields"
      dstProperty="primaryFields"
      dynamicProperties="primaryFields;foreignFields"/>
    <Item nodeName="Record"
      dynamicProperties="canAdd;canModify;canDelete;canSearch"/>
  </Items>
</File>
```

Item (Form / BusinessRecord / Database)

The `Item` element adds new dynamic properties to some existing node(s) in the Form or Business Record or Meta-schema, depending on `nodeName`.

The `Item` element has attributes.

Table 269: Item element attributes

Attribute	Options
<code>nodeName</code>	Corresponds to the node type name, such as "Relation".
<code>srcProperty</code>	Name of the property used as the source fields list, if the item is a Relation on page 977. Defines the source field while creating the query in application generation. The value of this property is the name of the dynamic property which holds source field. This dynamic property must be associated with relation through the <code>dynamicProperty</code> attribute.
<code>dstProperty</code>	Name of the property used as the destination fields list, if the item is a Relation on page 977. Similar to <code>srcProperty</code> it defines the destination field.
<code>dynamicProperties</code>	List of dynamic properties that apply to the item node.

Syntax

```
<BusinessRecord>
  <File extension="4rd">
    <DynamicProperties>
      <DynamicProperty name="foreignFields" type="FIELDS"
label="foreignFields"
      initialValue="" dynamicContent="srcFieldsContent"
      editorInfo="isDynamic:true" />
      <DynamicProperty name="primaryFields" type="FIELDS"
label="primaryFields"
      initialValue="" dynamicContent="dstFieldsContent"
      editorInfo="isDynamic:true" />
    </DynamicProperties>
    <Items>
      <Item
        nodeName="Relation"
        srcProperty="foreignFields"
        dstProperty="primaryFields"
        dynamicProperties="primaryFields;foreignFields" />
    </Items>
  </File>
</BusinessRecord>
```

creatables.conf elements

A listing of all available elements in the `creatables.conf` file.

The file is located in the template directory `GSTDIR/gst/bin/src/ag/tpl/dbapp`.

The XML schema for `creatables.conf` is `GSTDIR/gst/conf/schema/creatables-2.xsd`.

Note:

You can use `GSTDIR/gst/conf/schema/creatables.xsd` if you want to use the older format.

- [creatables.conf elements](#) on page 987
 - [Category](#) on page 988
 - [New, File, Directory, Wizard](#) on page 989

Creatables

The `Creatables` element is the root of the `creatables.conf` file.

The `Creatables` element contains two levels of `Category` elements.

Table 270: Creatables element attributes

Attribute	Options
<code>version</code>	Optional version information.
<code>xmlns:xsi</code>	Standard XML attribute for schema namespace.
<code>xsi:noNamespaceSchemaLocation</code>	Standard XML attribute for schema location.

Syntax

```
<Creatables>
  <Category attributes >
    <Category attributes >
      <New attributes />
    <File attributes />
    <Directory attributes />
```

```

    <Wizard attributes />
  </Category>
</Category>
</Creatables>

```

Child elements

The `Creatables` element may contain the following child elements:

1. One or more [Category](#) on page 988 elements.

Example

```

<Creatables version="2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.4js.com/ns/gst/2.50/
  creatables.xsd">
  <Category index="5" label="Design (MDA)" name="MDA" icon="document_4ba" >

    <Category index="40" label="Managed Code" name="ManagedCode">

      <New index="10"
        name="FDModuleForm"
        action="FDNew"
        label="Module Form (.4fdm)"
        icon="document_4fdm"
        description="Create an empty module in Form Designer"
        extension="4fdm"/>
    </Category>
  </Category>
</Creatables>

```

Category

`Category` elements group the actions and are used to organize actions in a way that are easy to find by a user.

The `Creatables` element contains two levels of `Category` elements. The first `Category` element corresponds to the **Categories** list in the **File >> New** dialog. The second `Category` element corresponds to the **Types** section in the **File >> New** dialog.

Table 271: Category element attributes

Attribute	Required	Options
index		Index indicates creatable order within a category.
name		Creatable name (internal id)
label		Creatable text.
icon		Icon in <code>GSTDIR/images/</code>

Syntax

```

<Creatables>
  <Category attributes >
    <Category attributes >
      <New attributes />
      <File attributes />
      <Directory attributes />
    <Wizard attributes />
  </Category>
</Creatables>

```

```

</Category>
</Category>
</Creatables>

```

Child elements

The `Category` element may contain the following child elements:

1. One or more `Category` elements.
2. One or more `creatable` elements.

Example: Managed Form File Type

```

<Creatables version="1.0">
  <Category index="5" label="Design" name="MDA" icon="document_4ba" >
    <Category index="40" label="Managed Code" name="ManagedCode">
      <Wizard index="10"
        name="FDModuleForm"
        action="FDNew"
        label="Module Form from Database (4fdm)"
        icon="document_4fdm"
        description="Create an empty module in Form Designer"
        extension="4fdm" />
    </Category>
  </Category>
</Creatables>

```

New, File, Directory, Wizard

A creatable is created with a New, File, Directory or Wizard element.

Table 272: Creatable element attributes

Attribute	Options	
index	Index of the creatable in a category.	New, File, Directory, Wizard
name	Creatable name (internal id). Required.	New, File, Directory, Wizard
action	See action attribute on page 990. Required for New and Wizard creatables.	New, File, Directory, Wizard
actionLabel	Label of the creatable when used as an action.	New, File, Directory, Wizard
icon	Creatable icon found. See Image directory structure on page 952.	New, File, Directory, Wizard
extension	Extension of the file to be created by the creatable.	New, Wizard, File
copy	Copies file(s) before opening. If used, user is asked to choose file path and name when executing the creatable.	File, Directory

Attribute	Options	
isTemplate	Specifies if file(s) should be opened using the template listed in the source attribute.	File, Directory
source	Path of source directory or source file used as the model for new file. Required.	File, Directory
masterFile	Specifies the master file (The name of a file in the directory).	Directory
open	List of files to open. This should specify files in the directory.	Directory
nameLabel	Label used in the File >> New dialog for the field where the user enters the name that will be given to the master file.	Directory

Syntax

```
<Creatables>
  <Category attributes >
    <Category attributes >
      <New attributes />
      <File attributes />
      <Directory attributes />
      <Wizard attributes />
    </Category>
  </Category>
</Creatables>
```

Example: Managed Form File Type

```
<Creatables version="1.0">
  <Category index="5" label="Design (MDA)" name="MDA" icon="document_4ba" >
    <Category index="40" label="Managed Code" name="ManagedCode">
      <Wizard index="10"
        name="FDModuleForm"
        action="FDNew"
        label="Module Form from Database (4fdm)"
        icon="document_4fdm"
        description="Create an empty module in Form Designer"
        extension="4fdm" />
    </Category>
  </Category>
</Creatables>
```

action attribute

The `action` attribute specifies the name of the action to activate when executing the creatable.

Action is required for `New` and `Wizard` creatables. If no action is provided for `File` and `Directory` creatables, `GSOpenFileAction` is used.

Note: Some file actions are unable to open some file types. If an action is used with an unsupported file type, the action will not be executed and the corresponding module error is displayed.

Table 273: action values

Name	Module	Content type	Description
GSOpenWithShell	Global	any	Open with external program.

Name	Module	Content type	Description
GSOpenFile	Global	any	Open file.
BAFileOpenAction	Business Application Modeling	application/genero-ba	Open BA diagram.
BROpen	Business Record	application/genero-br	Open Business Record.
CEOpen	Code Editor	any	Open in Code Editor.
CEOpenReadOnly	Code Editor	any	Open in Code Editor (read only).
CEOpenWithNoEdit	Code Editor	any	Open in Code Editor (no edit).
DBOpen	Meta-schema Manager	application/genero-db	Open Schema file.
DBImportSchemaFile	Meta-schema Manager	sch mime type based	Import sch file.
DBReloadSchema	Meta-schema Manager	application/genero-db	Update schema.
DBGenerateSCHFile	Meta-schema Manager	application/genero-db	Generate sch file.
DBGenerateDBCcreationFile	Meta-schema Manager	application/genero-db	Generation database creation script.
DBGenerateDBUpdateFile	Meta-schema Manager	application/genero-db	Generate database update script
FDOpen	Form Designer	application/genero-form	Open form.
FDFileImport	Form Designer	per mime type based	Import text form file (per).
PMOpenWorkspace	Project Manager	4pw mime type based	Open project.
RWLoad	Report Writer	4rp mime type based	Open report.
WPOpenUrl	WP		Open url.
WSOpen	Web Services	wSDL mime type based	View 4gl source.
WSUpdate	Web Services	mime type based	Update wSDL.

POINT and BLOCK reference

POINT and BLOCK sections that appear in the generated source code are defined within the template (Tcl) files. You can add additional POINT and BLOCK sections to the generated code.

Syntax

```
<POINT areaAttributes >...</POINT>
```

```
<BLOCK areaAttributes >...</BLOCK>
```

areaAttributes are standard XML attributes.

Table 274: areaAttributes

Attribute	Required / Optional	Usage
Name	required	Application's unique string identifier. Alphanumeric characters are allowed plus "_" and ".".
Aliases	optional	Space separated list of POINT / BLOCK names. This is used to rename a POINT / BLOCK. This value never appears in the generated source.
Status	optional	MODIFIED LOST
Action	optional	REVERT

Name and Aliases Attribute

If the name of a POINT or BLOCK is to be changed and source code has been generated, the name can be updated using the Aliases attribute. The Name attribute will contain the new POINT / BLOCK name. The previous name of the POINT / BLOCK is preserved in the Aliases attribute.

A POINT / BLOCK may be renamed several times, thus the Aliases attribute is a space-separated list of names. It is recommended that this space-separated list is sorted from the most recent to the oldest POINT / BLOCK alias.

Packaging, deploying, and distributing apps

When you create a package, you create a bundled file that contains all of the files needed to run the app via the Genero Application Server (GAS) or on a mobile device. Once the package file is built, it can be deployed to the GAS or to a device for testing.

Genero Studio eases the packaging and deployment of your app for testing by enabling you to build your package from within your project. See *Packaging, deploying, and distributing apps* in the *Genero Studio User Guide*.

Genero Archive (GAR) packaging

A Genero Archive (GAR) packages your application for deployment to the Genero Application Server.

To create a packaging node for your Genero Archive:

1. Add a package.
2. For the **Platform** property, select "**Genero Application Server**".

Note: The application configuration file (.xcf) should be included in the package. See [Create and apply a custom XCF for your Web application](#) on page 151.

A default manifest is generated for the package. If you want to create a custom manifest, see the *Genero Application Server User Guide*; the menu option **File >> New, Web / AS, Genero Archive MANIFEST** creates this template:

```
<?xml version="1.0" encoding="utf-8"?>
<MANIFEST xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.4js.com/ns/gas/3.00/mf.xsd">
  <DESCRIPTION>manifest description</DESCRIPTION>
  <TRIGGERS component="component_name">
    <DEPLOY>deploy_command</DEPLOY>
    <UNDEPLOY>undeploy_command</UNDEPLOY>
  </TRIGGERS>
  <RESOURCES>resources_directory</RESOURCES>
  <APPLICATION xcf="application.xcf" />
  <SERVICE xcf="service.xcf" />
</MANIFEST>
```

Packaging for a mobile device

Topics in this section are only relevant for packaging for a mobile device.

In terms of a mobile package, the package includes all files needed to run the program. It specifies the directory structure containing those files when the package is installed on the device.

Android packages have an `apk` extension, iOS packages have an `ipa` extension.

Packaging and Deploying topics:

- [What is packaging?](#) on page 994
- [Packaging process overview for a mobile device](#) on page 996
- [Package a mobile app](#) on page 996
- [Deploy a mobile app for testing](#) on page 998
- [Package and Directory nodes and properties](#) on page 999
- [Platform: Package and deploy rules](#) on page 1003

- [Distribute your app](#) on page 1005
- [Manage App updates](#) on page 1008

What is packaging?

Packaging prepares and packages all files required to deploy an app onto a mobile device.

In other words, packaging involves identifying what files are needed for the app, organizing those files, and creating a package file that can be deployed to the app. The specifics are summarized in the following paragraphs.

Directories involved

When you create a mobile app, and when you prepare for packaging, there are three directory structures that you must understand and manage.

Source directory structure

There is a directory structure containing the app sources. These are your source files - 4gl files, form definition files, image files, and so on.

Target directory structure

There is a directory structure containing the compiled binary files. As part of the compilation process, the compiled binaries are placed in target directories. In Genero Studio, you specify this location with the **Target directory** property. You specify this property for group nodes, application nodes, and library nodes.

Root directory structure

There is a directory structure containing the files needed on the mobile device. The mobile device or platform will have rules on what structure is allowed. The top-level directory in this structure is known as the Root directory (rootdir). In Genero Studio, the **Root directory** is defined as a property of the Package node. All file that needs to be deployed to the mobile device must sit under the Root directory.

The goal of packaging

To package your app, start by having an inventory of all files needed to run the app on the mobile device. Some of these files will be in your source directory structure (such as image files), while others will sit in your target directory structure (the compiled binary files).

You must also define the directory structure you will need to create in the Root directory for your mobile device. This may sound complicated, but in reality it is quite simple. For both the Android and iOS devices, you can place all of the files required into a single folder. If you examine the default packaging nodes provided when you create a new **BAM Mobile Project (.4pw)**, you see that all files end up in the directory `$(ProjectDir)/bin`, which is the packaging Root directory. You could build a more complex Root directory structure, however it is not necessary.

Once you have your inventory of required files and your plan for the directory structure on your device, you can build your packaging node.

In the Package node, you define the Root directory and specify which platform the package is for.

Under the Package node, you create a Directory node for each directory you need to retrieve files from. You will have multiple Directory nodes, where each Directory node serves a purpose: locating the binaries, database files, fglprofile, images and so on.

Important: It is not sufficient to physically put a file in the Root directory on disk to have it included in the final package. Any file you want to include in the package must be listed as a source file in one of the directory nodes.

The Source directory indicates where the files exist prior to packaging, the Destination directory indicates where the files will be located in the package. The Destination directory must be located within the Root directory. In the Directory node, the Included files and Excluded files properties tell which files from the Source directory to include or exclude, based on filtering criteria (such as the filename or file extension).

During the packaging process, the files are copied from the source directory to the Destination directory, if these directories differ.

Note: If you examine the default packaging nodes provided when you create a new **BAM Mobile Project (.4pw)**, you will see many Directory nodes where both the Source and Destination directories specify `$(ProjectDir)/bin`. Having the Source and Destination specify the same directory allows for packaging optimization, by not requiring files to be copied from one directory to another. Files are only copied when they are moving to a different directory. You can define separate Source and Destination directories, however the packaging process would not be optimized as a complete directory copy would be required each time the package is built.

Have your program move files into a read-write (writable) directory

If you create a flat file to hold all the files required by the app, a flat file is created on the mobile device. This is what is done by the default packaging nodes when you create a new **BAM Mobile Project (.4pw)**.

For your Android applications, this works out-of-the-box. After the package is deployed to the device, it is unpacked into a single directory on the device. This is a read-write directory, which means that any file that need to be writable (such as a database file) can be updated by the app.

Warning: If you redeploy an Android application, ALL files are overwritten, to include files such as a database file. You must take this into account as you plan your application upgrades, and handle any upgrade strategy in your app.

For your iOS applications, however, the directory created and holding the app files is a read-only directory. If there exists writable files that need to be updated by the app (such as a database file), those files must be placed into a read-write directory on the device. Moving files from the read-only directory to a read-write directory is not something handled by packaging. You must handle it within your app.

You are provided with two APIs that allow you to reference the underlying directories transparently:

- The `base.Application.getProgramDir()` method returns the base program directory, storing your compiled files, an initial database file, and so on. On an iOS device, this is a read-only directory.
- `os.Path.pwd()` defines a writable directory for holding writable files, such as an error log or the user database.

At deployment, when your application initially starts, we recommend that you copy the writable files from the application directory (using the function `base.Application.getProgramDir()`) to the current working directory (using the function `os.Path.pwd()`).

For an example, open the **OfficeStoreMobile** demo project, open the **OrdersApp.4gl** intermediate file, and look for the `OrdersApp_install` function. In this function, the `os.Path.copy` method is used to copy a database file from the read-only source directory to the read-write destination directory.

Warning: Take care that you do not use the same file name for a read-only resource and a read-write resource. Using the same name for both can lead to problems.

Create the user database in a writable directory

In addition to simply copying writable files, one of the first things your app needs to do is create the user database. This database may use the initial database file when creating the user database, or it may create one with a different schema for the user.

For an example, open the **OfficeStoreMobile** demo project, open the **OrdersApp.4gl** intermediate file, and look for the `OrdersApp_install` function. In this function, the `os.Path.copy` method is used to copy a database file from the read-only source directory to the read-write destination directory.

App initial deployment versus App upgrade

The example provided by the `OrdersApp.4gl` example needs to be evaluated in terms of initial installation and in terms of future upgrades to the application. This example solves how to initially place the file in the read-write directory. It does NOT cover an upgrade strategy. The developer needs to ensure that any new files need to be written to the read-write directory are written, that files that should not be overwritten are not overwritten, and that files that need to be updated or merged or replaced are handled. See [Manage App updates](#) on page 1008.

Packaging process overview for a mobile device

Creating a package gathers and organizes all required files for a single mobile app.

If you created your project using the **BAM Mobile Project (.4pw)** or **Mobile Project (.4pw)** options, default package nodes were created for both Android and iOS devices. If you need to create a package from scratch, the steps include:

- Create a Package node in your project and set its properties.
- Create Directory nodes in your Package node and set their properties.
- Confirm the Package Rules for the Platform defined for the Package node.
- Right-click on the Package node and select Build. A package is built.
- Right-click on the Package node and select Deploy. A package is moved to the connected device or emulator.
- Distribute your app to users. See [Distribute your app](#) on page 1005.

Package a mobile app

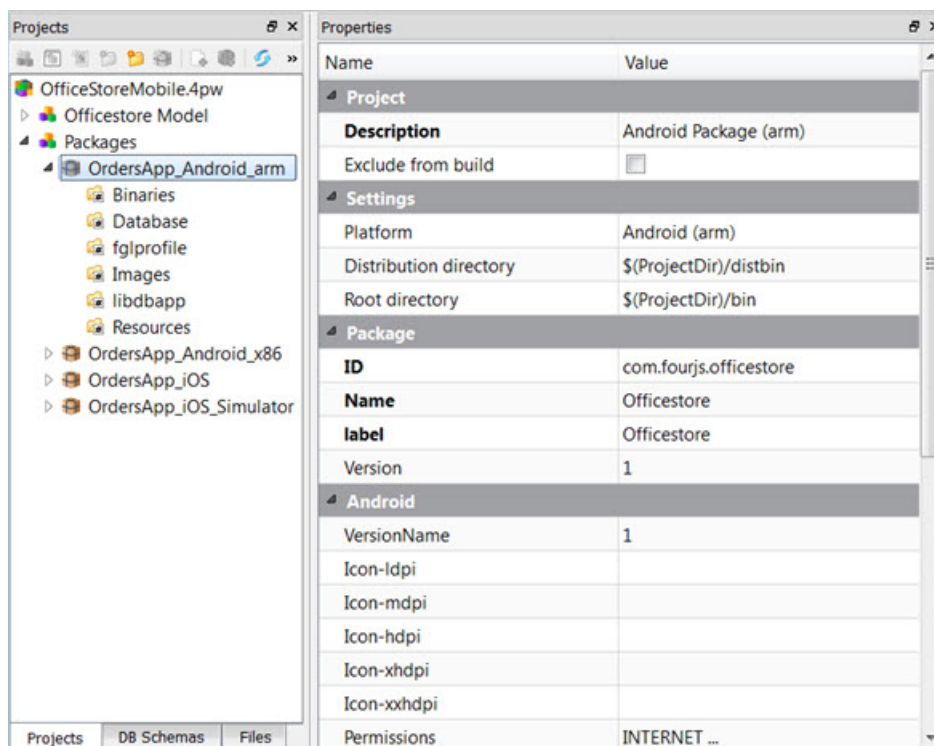
Follow this procedure to create a package for your mobile app.

If you created your project using the **BAM Mobile Project (.4pw)** or **Mobile Project (.4pw)** options, default package nodes were created for both Android and iOS devices. Even with these packaging nodes, you should still read through this procedure to add any additional directories and validate all properties.

1. Launch Genero Studio.
2. Open your project.
3. Make a list of all the files needed to run your app and the full path to where these files currently reside.
4. Right-click on the **Group** node and select **New Package**.
5. Define the [package properties](#).

You can accept many of the default values. You should, however, provide details for the properties listed under the **Package** group.

Figure 410: Package properties



6. Create a New Directory in the Package node for each physical directory on disk that contains files to be included in the package.

Right-click on the Package node and select **New Directory**. Define the properties for each directory added.

In the **Source directory** property, you must specify the directory where the file(s) will come from. In the **Destination directory** property, specify the directory where you want the file copied to. This directory must be contained in the Root directory.

Note: In the default project that is set up when you create a new **BAM Mobile Project (.4pw)**, both of these properties are set to `$(ProjectDir)/bin`. If the source and destination directories are the same, the packaging process does not have to perform the copy, resulting in an optimization of the packaging process.

If you need to include or exclude files from the source directory, you would add the criteria to the **Included files** and/or **Excluded files** properties.

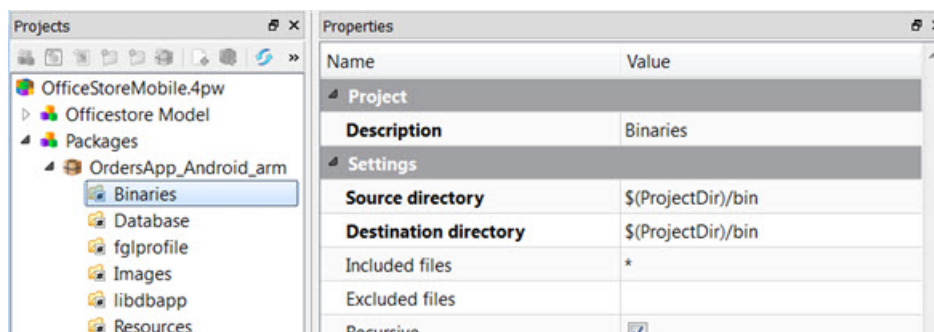


Figure 411: Directory properties

7. Confirm the **Package rules**. Right-click on the Package node and select **Edit Package Rules**.

Note: You will typically not have to modify the package rules.

- Right-click on the package node and select **Build**. This action builds the package according to the package rules. Progress of the packaging will appear in the output panel.

Important: You need an internet connection for the first time you build an Android package. During this first build, an automated process will download and install Gradle with all necessary extensions into a directory in your user directory. Gradle is a project automation tool, find out more about Gradle at <http://www.gradle.org>.

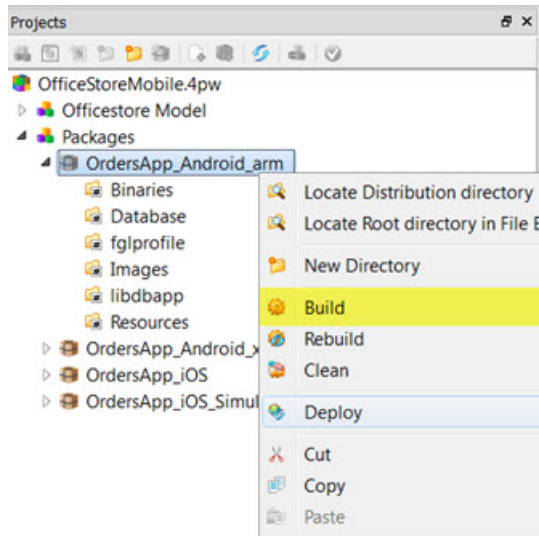


Figure 412: Build package

- Confirm that the package is on your local file system.
- Validate this package on a device by transferring it to the device, installing it on the device, and running it on the device. To do so, follow the instructions to [Deploy a mobile app for testing](#) on page 998.

Troubleshoot Android packaging issues

Here are troubleshooting tips for issues you may encounter when packaging your Android app.

Unexpected warning regarding a timestamp

If you are packaging with Java version 1.7.51 or later, you may receive this message:

```
Warning: No -tsa or -tsacert is provided and this jar is not timestamped.
Without a timestamp, users may not be able to validate this jar after
the signer certificate's expiration date (YYYY-MM-DD) or after any future
revocation date.
```

This warning appears because the `.apk` is signed without a timestamp. A timestamp is not required for Android. The debug certificate validity period is very long; its expiration should not be a problem.

Deploy a mobile app for testing

Follow this procedure to move your package to a connected device or emulator.

This procedure assumes you have [packaged an app](#).

- If deploying to a device, plug the device into a USB port.
 - If necessary, update the drivers for your device.
 - For Android, you can check that your computer recognizes your device by selecting **Tools >> Android Tools >> List Devices**. See [Configuring Genero Mobile for development](#) for full configuration information.
- In Genero Studio, select the correct Genero configuration.
 - For example, select **Android ARM** for Android devices.

3. Right-click on the package in your project and select **Deploy**.

The Deploy action executes the commands in the Deploy rule for the platform specified for the package node. Progress of the deploy action appears in the output panel.

You can test this with the **OfficeStoreMobile** project. Open the project and deploy the corresponding package in the Packages group.

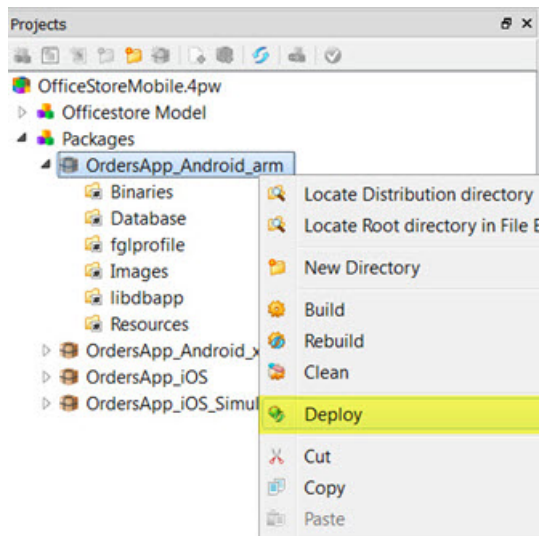


Figure 413: Deploying one of the example packages

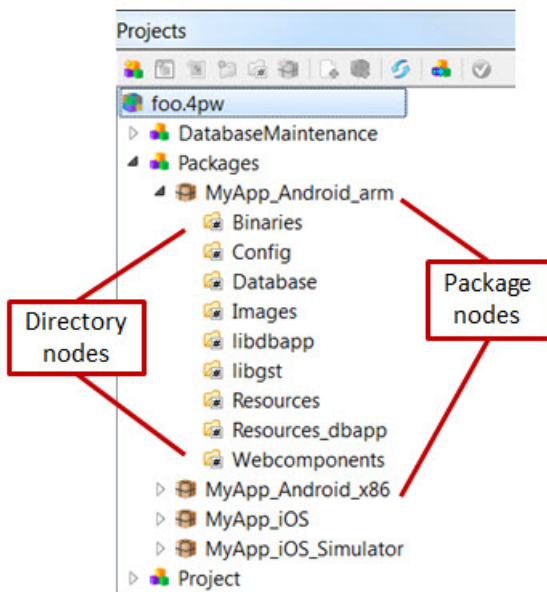
The app will appear on the device or emulator.

4. Tap the app icon to launch the app.

Package and Directory nodes and properties

A package node defines the packaging instructions for the app. A directory nodes specifies one or more files to be included in the package.

Package and Directory Nodes



Package nodes

Each Package node defines the information needed to build and deploy the app. Similar to an Application node, a Package node can have dependencies set to other Libraries or Application in the project. Package nodes can be created within a Group node in a project. Build and Deploy actions are available by right-clicking on the Package node. Each Package node can have Directory nodes to further organize the package.

Table 275: Default Package nodes

Node	Description
MyApp_Android_arm	Default package setup for an Android package to be deployed to an Android physical device.
MyApp_Android_x86	Default package setup for an Android package to be deployed to an Android Virtual Device.
MyApp_iOS	Default package setup for an iOS package to be deployed to an iOS physical device.
MyApp_iOS_Simulator	Default package setup for an iOS package to be deployed to an iOS Simulator.

Table 276: Package Node properties

Group	Property	Description
Project	Description	Optional textual description of the package.
Project	Exclude from build	Excludes the node from the build process.
Settings	Platform	Platform used to package and deploy. See Platform on page 1003.
Settings	Distribution directory	Directory where the output files should be generated. The output files refer to the packaged deliverables -- the <code>.apk</code> file for Android devices or the <code>.ipa</code> file for iOS devices. The last step of the packaging process would be to put all of the Root directory files into an <code>.apk</code> or <code>.ipa</code> file and to put that final file into the Distribution directory.
Settings	Root directory	Directory from which files will be packaged. The package contents are organized using the path relative to this directory.
Package	ID	Unique id.
Package	Name	Package name. Also used in package file names.
Package	label	Label displayed to user.
Package	Version	Optional version number.
Android	VersionName	Optional version name.
Android	Primary color	Define the primary color, or the main color used in the app (Android 5.0 and later). For more information, see the <code>gmabuildtool</code> topic in the <i>Genero Business Development Language User Guide</i>
Android	Primary dark color	Define the primary dark color, used for the status bar and the navigation bar (Android 5.0 and later). For more

Group	Property	Description
Android	Accent color	information, see the <code>gmabuildtool</code> topic in the <i>Genero Business Development Language User Guide</i> Define the accent color, used for widgets and table lines (Android 5.0 and later). For more information, see the <code>gmabuildtool</code> topic in the <i>Genero Business Development Language User Guide</i>
Android	Action bar text color	Define the action bar text color, used as the foreground color for the texts in the action bar (Android 5.0 and later). For more information, see the <code>gmabuildtool</code> topic in the <i>Genero Business Development Language User Guide</i>
Android	Icon-ldpi (36x36 px) Icon-mdpi (48x48 px) Icon-hdpi (72x72 px) Icon-xhdpi (96x96 px) Icon-xxhdpi (144x144 px)	Application icon. Icons must be in PNG format. See Iconography and Devices and Displays on the Android Developer site for more information about icons and their use with Android apps.
Android	Permissions	Android permissions requested during package install, such as request to access device camera.
iOS	Launch Screen (320x480 px) Launch Screen (640x960 px) Launch Screen (640x1136 px) Launch Screen (750x1334 px) Launch Screen (1334x750 px) Launch Screen (1242x2208 px) Launch Screen (2208x1242 px) Launch Screen (768x1024 px) Launch Screen (1024x768 px) Launch Screen (1536x2048 px) Launch Screen (2048x1536 px)	Image used for the iOS Launch Screen. The iOS Launch Screen displays when your app is first run and before the first window displays, allowing the user to see your app responding while startup tasks are completing. See Launch Images on the iOS Developer Library site for more information about icons and their use with iOS apps. <ul style="list-style-type: none"> • Launch Screen (320x480 px) is the launch screen for iPhone • Launch Screen (640x960 px) is the launch screen for iPhone 4 • Launch Screen (640x1136 px) is the launch screen for iPhone 5 • Launch Screen (750x1334 px) is the launch screen for iPhone 6 (portrait) • Launch Screen (1334x750 px) is the launch screen for iPhone 6 (landscape) • Launch Screen (1242x2208 px) is the launch screen for iPhone 6+ (portrait) • Launch Screen (2208x1242 px) is the launch screen for iPhone 6+ (landscape) • Launch Screen (768x1024 px) is the launch screen for iPad (portrait) • Launch Screen (1024x768 px) is the launch screen for iPad (landscape) • Launch Screen (1536x2048 px) is the launch screen for iPad (portrait) • Launch Screen (2048x1536 px) is the launch screen for iPad (landscape)
iOS	Launch Screen XIB	iOS 8.0 introduced a new launch screen file format as a replacement to the PNG images. This format is created using Xcode 6.0 or later.

Group	Property	Description
iOS	Icon (29x29 px)	<p>To create the file, open Xcode and select File>>New>>File...>>User Interface>>Launch Screen. Design the launch screen using Xcode.</p> <p>For more information, see Apple's documentation on launch images.</p> <p>Application icon. Icons must be in PNG format. See Icon and Image Sizes and App Icon on the iOS Developer Library site for more information about icons and their use with iOS apps.</p> <ul style="list-style-type: none"> • Icon (29x29 px) is a 29x29 pixel icon (used for Settings) • Icon (40x40 px) is a 40x40 pixel icon (used for Settings) • Icon (58x58 px) is a 58x58 pixel icon (used for Settings) • Icon (60x60 px) is a 60x60 pixel icon (used for Settings) • Icon (76x76 px) is a 76x76 pixel icon (used for the Home Screen) • Icon (80x80 px) is a 80x80 pixel icon (used for the Home Screen) • Icon (87x87 px) is a 87x87 pixel icon (used for the Home Screen) • Icon (120x120 px) is a 120x120 pixel icon (used for the Home Screen) • Icon (152x152 px) is a 152x152 pixel icon (used for the Home Screen) • Icon (180x180 px) is a 180x180 pixel icon (used for the Home Screen)
	Icon (40x40 px)	
	Icon (58x58 px)	
	Icon (60x60 px)	
	Icon (76x76 px)	
	Icon (80x80 px)	
	Icon (120x120 px)	
	Icon (152x152 px)	
	Icon (180x180 px)	

Directory nodes

The Directory nodes specify the source and destination directories, which files to include and which files to exclude as they move from source to destination, and whether to recursively search in the sub directories. Default Directory nodes are created when creating a mobile project. The properties of the default directories can be modified and additional directories can be added as needed. Any physical directory that includes files that need to be included in the package must have a virtual directory created. Some of the default directory nodes correspond to the default project structure.

Table 277: Default Directory nodes

Directory	Description
Binaries	Location of binary files such as the compiled modules (42m files).
Config	Location of configuration files. By default, <code>fglprofile</code> , is packaged to provide the database driver and source information.
Database	Location of database files. By default, <code>db</code> files are specified to be included as SQLite is the supported database.
Images	Location of image files. By default, <code>png</code> and <code>jpg</code> files are specified to be included as these are the supported image types.
libdbapp	Location of the Business Application Modeling template libraries.
libgst	Location of the libraries used with the Database Generation script.

Directory	Description
Resources	Location of external files such as action defaults file (4ad) and style file (4st).
Resources_dbapp	Location of Business Application Modeling template resource files, dbapp.4ad and dbapp.4st.
Webcomponents	Location of web component files.

Table 278: Directory Node properties

Group	Property	Description
Project	Description	Description of directory.
Settings	Source directory	The path to the directory containing the resources.
Settings	Destination directory	Directory where files are put during the packaging process. Must be within the package Root directory.
Settings	Included files	Filter of the files to include in the resources. For example, * to include all files, *42f *42m to include only files that match these types. Important: Separate filters with a space, not a comma.
Settings	Excluded files	Filter of the files to exclude from the resources.
Settings	Recursive	Check to search the files recursively in the sub directories.

Platform: Package and deploy rules

A platform includes a package rule and a deploy rule. These rules are the commands used to create and deploy a package file.

Platform

A platform is similar to the concept of building and running an app based on a set of build, link and execution rules, but instead the rules are for packaging and deploying. The platform corresponds to the Platform property for the Package node in the project, defining the package and deploy rules to use.

There are currently four platforms for Genero Mobile:

Android (arm)

For Android, use the appropriate processor depending on the setup of your Android Virtual Device (AVD) or your physical device hardware. To avoid performance issues, we recommend that you configure your AVD with an x86 processor and Intel Hardware emulation enabled (HAX). When you create the default AVD using the Genero Studio menu option, the AVD has an x86 processor.

Android (x86)

See [Android \(arm\)](#).

iOS

For an iOS physical device.

iOS Simulator

For an iOS simulator.

Package rules

The package rule contains a list of files that are generated by the package rule (output files) and a list of dependencies to determine if the output files are up to date. The Package Rules can be viewed by right-clicking on the **Package Node** and selecting **Edit Package Rules**.

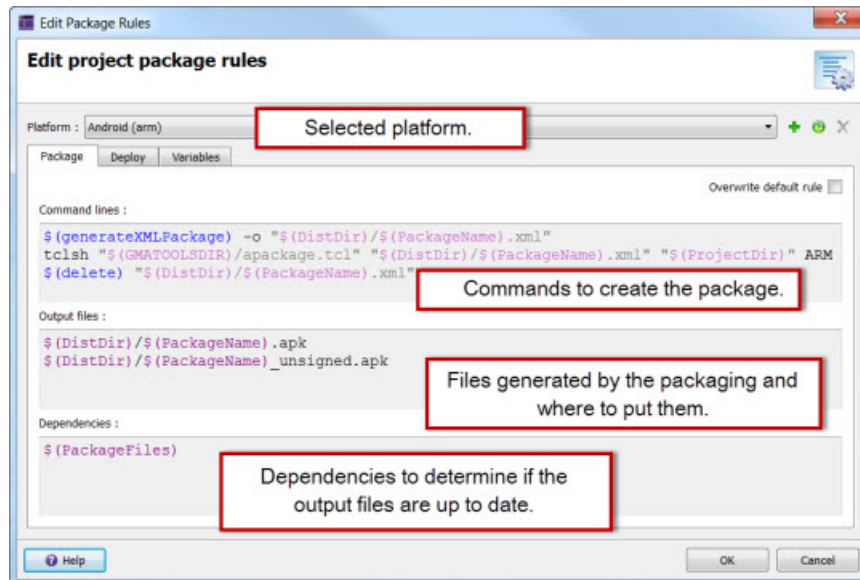


Figure 414: Example Package Rule

Deploy rules

A deploy rule runs a script to deploy the package to a connected device or emulator.

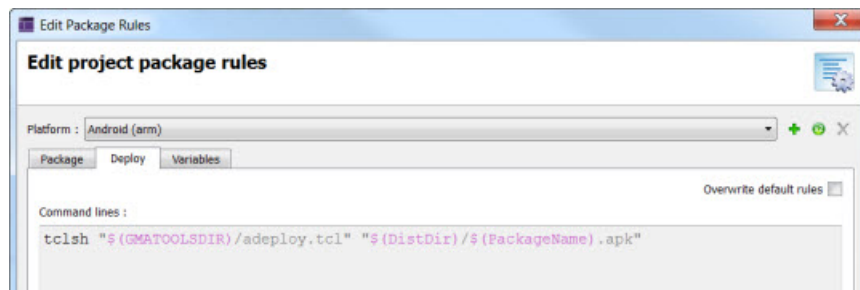


Figure 415: Example Deploy Rule

Commands and variables

These commands and variables are specific to package and deploy rules, but other predefined node variables such as `$(ProjectDir)` and `$(InstallDir)` are also valid. See the *Predefined node variables* topic in the *Genero Studio Developer Guide* for a list of node variables.

\$(delete)

This command, available in a package rule, deletes the specified file.

\$(DistDir)

Variable for the path of the destination directory of the current package node.

\$(generateXMLPackage)	This command, available in a package rule, generates an XML file that describes the contents of the package.
	<pre>\$(generateXMLPackage) -o outputFile.xml</pre>
\$(GMATOOLSDIR)	Environment variable for the path to the tools that support Genero Mobile for Android.
\$(GMITOOLSDIR)	Environment variable for the path to the tools that support Genero Mobile for iOS. Set
\$(PackageFiles)	Variable for the list of files in the package.
\$(packageId)	Variable for the package id.
\$(PackageName)	Variable for the name of the current package node.

Distribute your app

After creating your app package, you are responsible for getting the app into the hands of your users.

- [Distribute an iOS app](#) on page 1005
- [Distribute using Over-The-Air installation](#) on page 1005
- [Distribute an Android app](#) on page 1007
- [Distribute your app through Google Play](#) on page 1007
- [Distribute using other tools for testing](#) on page 1008

Distribute an iOS app

Apple provides two options for distributing your app to iOS devices.

With Genero Mobile, you are able to create a signed iOS app package (with an `.ipa` extension).

At this point, you have two options offered by Apple for distributing your apps.

- To distribute your app using the App Store, you must join the [Apple iOS Developer Program](#).
- To distribute proprietary, in-house iOS apps to your employees, you must join the [iOS Developer Enterprise Program](#).

These are two separate programs. The Apple ID that you get with your iOS Developer Enterprise Program will not let you submit your app to the App Store. You will need to join both programs (and get two Apple IDs) if you need to submit your app to the App Store *and* distribute proprietary, in-house iOS apps to your employees.

Distribute using Over-The-Air installation

Follow this procedure to distribute your app in-house to iOS devices via an over-the-air (OTA) installation.

This procedure assumes you have prepared to [package the app](#) for iOS. As you are packaging an iOS app, this can only be done from a Mac OS development machine. You must be a member of the [iOS Developer Enterprise Program](#).

Note: If you are not a member of the iOS Developer Enterprise Program, yet want to distribute your app in-house for testing purposes only, see [Distribute using other tools for testing](#) on page 1008.

This procedure allows you to create and distribute an iOS package using an *Over the Air (OTA)* installation.

1. Set the `OTA_URL` environment variable to a URL. When the package is created, a `packagename.plist` file is generated. A URL is displayed in the output that can be used for OTA install.

2. Upload the `plist` file, the `ipa` package, `image57x57.png`, and `image512x512.png` to the server in the directory corresponding to the given URL.

Important: The Web server must be an HTTPS server.

3. Send the URL to the iOS device by mail or SMS, or add it to an HTML document on a web server.
4. On the iOS device, click on the URL.
The app installs on the device.

Submit an iOS app to the AppStore

If you want your users to download their app from the Apple AppStore, you must submit your app for approval.

It is assumed you have Apple Developer Program membership, which is required to request, download, and use signing certificates issued by Apple. See <http://developer.apple.com/programs/ios/>.

This procedure provides you with the steps for submitting an app to the AppStore. It includes steps for:

- Creating an App ID
 - Obtaining a distribution certificate
 - Creating an iOS Distribution Provisioning Profile
 - Delivering an app to the AppStore
1. Go to <https://developer.apple.com/> web site and sign in to the **Member Center** using your Apple ID.
 2. Go to the **iOS** platform page and register an iOS App ID by providing an app name and an App ID suffix. There are two options for setting the App ID Suffix:
 - By setting Explicit App ID
 - By setting a WildCard App ID

Note: An Explicit App ID matches a single app and a WildCard App ID matches multiple apps; both methods are explained in the iOS App developer site.
 3. Create an iOS Distribution Certificate:
 - a) Go to the Certificates page and select the plus button +
 - b) Select App Store and Ad Hoc under the Production section
 - c) Follow the instructions to create your Certificate Signing Request (CSR) and upload it
 - d) Download your iOS Distribution Certificate and open it in `Keychain Access` (Apple's password-management application). In `Keychain Access` you should see your iOS Distribution Certificate with the private key.
 4. Add a new iOS Distribution Provisioning Profile
 - a) In the <https://developer.apple.com/> web site go to the Provisioning Profiles page and select the plus button (+) to add a new profile
 - b) Select the App Store
 - c) Select your App ID (previously registered)
 - d) Select your iOS Distribution Certificate
 - e) Set a profile name
 - f) Generate and download your iOS Distribution Provisioning Profile
 5. In Genero Mobile's configuration window
 - a) Set the `IDENTITY` to your iOS Distribution Certificate's User ID
 - b) Set `PROVISIONING_PROFILE` variables to the path to your Distribution Provisioning Profile (*.mobileprovision)
 - c) On your project, edit the `ID` property of the iOS package you wish to submit, to match the suffix of the iOS app ID previously created
 - d) Build your iOS Package
 6. Create your application on iTunesConnect.

Refer to Apple's documentation for [information about using iTunes Connect](#)

Note: Only an agent of the Apple iOS Developer Program can connect to iTunesConnect

- a) From Genero Studio, select **Tools iOS >> iOS Tools>>App Store deployment>>Open iTunesConnect**. (This opens the <http://itunesconnect.apple.com> web page.)
 - b) Select My Apps and the option to add a New iOS App
 - c) In the form, choose your iOS App ID from the Bundle ID field
 - d) Fill in the other fields according to your app.
 - e) Select Create
7. Upload the app for submission to Apple using Xcode's Application Loader
- a) From Genero Studio, select **Tools >> iOS Tools >> iOS App Store deployment >> Launch Application Loader**. This opens Xcode's Application Loader application. Refer to Apple's documentation for [information about using Application Loader](#)
 - b) Select Deliver your App
 - c) Select the Choose button and select the iOS App Store Package (.ipa) file generated on building your app with Genero Mobile. Your application will now be validated. If there are errors, they will be displayed at the end of the validation process.

Important: Only an agent of the Apple iOS Developer Program can submit the app.

Distribute an Android app

Android has several methods for distributing your app to Android devices.

With Genero Mobile, you are able to create an Android app package (with an `apk` extension).

As an open platform, Android offers several methods for distributing your app: via email, via a Web site, or via the Google Play Store. See [Open Distribution](#) for more details.

Distribute your app through Google Play

To reach the broadest possible audience, distribute your Genero Mobile for Android (GMA) app through the Google Play marketplace.

Before you begin, you must have a Google Developer account. To sign up, visit <https://play.google.com/apps/publish/signup/>. To see the agreement, visit <https://play.google.com/about/developer-distribution-agreement.html>

Create an APK package that meets the requirements of Google Play, and load it to the Google Play Store.

1. Create the APK for the app.

The APK must meet specific requirements in order to upload to the Google Play Store.

Set the environment variables dedicated to application signing. These environment variables must be defined in Genero Studio configuration settings, typically in the Android environment set:

- `KEYSTORE_PATH` - Path to the key store
- `KEYSTORE_PW` - Key store password
- `KEYSTORE_KEY_ALIAS` - Entry name for the key in the key store
- `KEYSTORE_KEY_PW` - Key password in the key store

By default, these variables are set to use a dummy key (`androiduserkey:android`). Change to package with another key.

Note: We recommend you use the default key store for development, in order to avoid distributing your company private signing key to all developers. Set up a separate secured environment for signing with your company key.

2. Open the [Google Play Developer Console](#) and follow the process to add your app to the store.

Distribute using other tools for testing

After creating your app package, there are various third-party tools to distribute your app for testing purposes only.

Additional tools are available to assist you in beta testing your apps, such as [TestFlight](#) or [HockeyApp](#) (which offers Android support) . Such tools provide an additional method for beta testing your app, and includes such features as Over-The-Air distribution and the gathering of feedback from testers. They are not, however, alternate solutions for the final distribution of your apps to your end users.

Manage App updates

When an new version of an app installs on a device, you might want to keep all or part of the data created with the previous version.

It is the role of the deployed program to manage the updates.

App Update Strategy

Detect if a previous app is installed. Each deployed app should store a file (named `version.txt` for this example) that contains the version of the deployed app. This file allows the app to detect if an older version of the app is installed by checking for the existence of the `version.txt` file. By reading the file contents, the app can retrieve the exact version of the previously deployed app.

When no previous app is detected, perform a fresh install. This might involve these tasks:

- Create `version.txt`.
- Create configuration files.
- Create and populate the database.
- Copy read-only files to the writable document area.

When an previous install of the app is detected and its version identified, perform an upgrade. This might evolve these tasks:

- Update `version.txt` to the new version number.
- Convert existing configuration files to a new file format.
- Upgrade the database schema while keeping the data.
- Delete obsolete files.
- Copy read-only files to the writable document area.

Tip: Having a clear separation between the initialization data (in the read-only directory) and the current user data (in the read-write directory) can help you manage your upgrades.

You are provided with two APIs that allow you to reference the underlying directories transparently:

- The `base.Appliation.getProgramDir()` method returns the base program directory, storing your compiled files, an initial database file, and so on. On an iOS device, this is a read-only directory.
- `os.Path.pwd()` defines a writable directory for holding writable files, such as an error log or the user database.

When the version of the previous install is an older version of the app, a strategy might be to run in sequence all upgrade scripts from the detected version to the latest. For example, when installing v1.5 on a device where v1.3 is the previous version, the upgrade script from v1.3 to v.4 will be executed first, then the upgrade script from v1.4 to v1.5.

This strategy involves storing modified data outside of the directory folders where the app installs, to prevent user data from being overwritten during install. This allows the programmer to read old data and do the appropriate steps to (eventually) convert them to a newer format.

Tools

The **Diff Schema** and **Generate Database Update Script** tools in Genero Studio provides database migration scripts from a previous database schema to a newer database schema.

- **Diff Schema** identifies the differences between two schemas
- **Generate Database Update Script** takes the file created using **Diff Schema** and generates an upgrade script. This database upgrade script can be incorporated in the app sources to manage the database upgrade.

These scripts can be called in sequence to migrate from an older version to the current.

Testing updates

App updates should be tested in a real life environment.

- Install the previous production app on a device.
- Use the app such that it stores user data.
- Deploy the new version of the app.
- Verify the upgrade works as expected.

For more information, see [Technical Note TN2285: Testing iOS App Updates](#) in the *iOS Developer Library*.

Upgrading

These topics talk about what steps you need to take to upgrade to the next release of Genero Studio, and allows you to identify which features were added for a specific version.

- [New Features of Genero Studio](#) on page 1010
- [Upgrade Guides](#) on page 1022
- [Migrating to a new BAM template set](#) on page 1027

New Features of Genero Studio

These topics provide a look back at the new features introduced with each release of Genero Studio.

- [What's new in Genero Studio, v 3.00](#) on page 11
- [What's new in Genero Studio, v 2.51](#) on page 1014
- [What's new in Genero Studio, v 2.50](#) on page 1014
- [What's new in Genero Studio, v 2.41](#) on page 1021
- [What's new in Genero Studio, v 2.40](#) on page 1021

What's new in Genero Studio, v 3.00

This publication includes information about new features and changes in existing functionality.

Important: Please read [What's new in Genero Studio, v 2.51](#) on page 1014, for a list of features that were introduced with the Genero Mobile 1.0 release.

These changes and enhancements are relevant to this publication.

Table 279: General, Version 3.00

Overview	Reference
Genero Studio now supports Genero, Genero Report Writer for BDL, and Genero Mobile from a single installation.	N/A
Genero Studio now supports the Genero Web Client for Javascript (GWC-JS).	See Web: GAS/GWC configurations on page 147.
Genero Studio now supports connecting to the application server using HTTPS.	See Web: GAS/GWC configurations on page 147.
Generate a Genero archive (GAR) from Studio, for deployment to the GAS.	See Genero Archive (GAR) packaging on page 993
Genero Studio gives the ability to locate a document in the File Browser, in a BA diagram, or in the System File Browser (the file browser of the operating system). The new System File Browser feature facilitates the use of system file explorer integrated tools, such as SVN or Git integration.	See Locate a file (starting at Project Manager) on page 353 or Locate a file (starting at File Browser) on page 501.
New configuration option for GAS, allowing the ability to run and debug Web services from Genero Studio.	See Web: GAS/GWC configurations on page 147.

Table 280: Project Manager, Version 3.00

Overview	Reference
The properties Web Service and Web Service URL suffix have been added for the Application node, allowing the ability to run and debug Web services from Genero Studio.	See Project Manager node properties on page 364.

Table 281: DB Explorer, Version 3.00

Overview	Reference
DB Explorer module introduced to view and modify data in database tables and to test SQL query results. With this tool, you can right-click on forms, reports and Web services to view the data.	See DB Explorer on page 332.
DB Explorer expands support of SQL commands, in addition to SELECT, INSERT, UPDATE, and DELETE.	See DB Explorer on page 332 and Write a SQL query by hand on page 338.

Table 282: Meta-Schema Manager, Version 3.00

Overview	Reference
Enhanced schema view displays database schema modifications at a glance.	See Viewing a meta-schema on page 300.
Mouse over items in the schema for more detail, to include a summary of schema changes, primary key column details, and more.	See Viewing and manipulating a meta-schema .
Reorder columns using drag-and-drop.	See Viewing and manipulating a meta-schema .
Move columns to another table using drag-and-drop.	See Viewing and manipulating a meta-schema .
HTML meta-schema documentation provides details of all database objects and facilitates global schema review.	See Generate meta-schema documentation on page 305.
Toggle label display shows or hides foreign key names in the diagram.	See Viewing and manipulating a meta-schema .

Table 283: Genero Mobile, Version 3.00

Overview	Reference
You can debug an application deployed to a mobile device. With this new feature, the application is running on the mobile device and the Graphical Debugger is able to attach to the process.	See Debug a mobile application on page 504
The <code>DBAPP_MOBILE</code> environment variable provides warning messages regarding features not supported by mobile devices during the compilation of applications generated by the Business Application Modeler.	See DBAPP_MOBILE on page 268.

Table 284: Business Application Modeler, Version 3.00

Overview	Reference
Publish JSON Web services via the Business Application Modeler.	See JSON Web services on page 220.
SOAP Web services enhanced with XML and XSD Schema Serialization attributes.	See Webservice entity on page 217.

Table 285: Code Editor, Version 3.00

Overview	Reference
Code Editor supports a horizontal view in the Diff tool, in addition to the vertical view of previous versions.	See Using the Diff tool on page 380.

Table 286: Form Designer, Version 3.00

Overview	Reference
Support for new Form properties: <code>keyboardHint</code> , <code>completer</code> , <code>wantFixedPageSize</code> , <code>action</code> , <code>Disclosure Indicator</code>	See Properties list on page 459.
Support for new <code>DateTimeEdit</code> widget.	See DateTimeEdit on page 425.

Table 287: Search, Version 3.00

Overview	Reference
Search Results pane displays an improved search view, to include previous and current search results organized as a collapsible tree.	See The Search Results view on page 393.

Table 288: Genero Report Writer, Version 3.00

Overview	Reference
Genero Report Viewer for HTML5 provides a browser equivalent of the Genero Report Viewer.	See fgl_report_selectDevice on page 626.
A command-line utility checks and upgrades report design documents (<code>.4rpt</code>) files in batch.	See Upgrading reports from prior versions on page 837.
Report templates provide a wizard-based method for creating report design documents (<code>.4rpt</code>) from a generic report design. The wizard allows you to bind repeating sections, add fields, and bind placeholders and parameters from a data schema, in order to create a stand-alone report design document. A library of report templates have been provided, and you can create your own templates. A template expansion mechanism is available as a command line tool, usable from applications for generic reports.	See Report templates on page 840.
The report engine now limits the number worker threads in distributed mode, to prevent memory exhaustion in times of critical load. Change the default value (25 threads) with the environment variable <code>GRE_MAX_CONCURRENT_JOBS</code> .	See GRE_MAX_CONCURRENT_JOBS on page 858.
You can now configure the default output directory for the Genero Report Engine with the <code>GREOUTPUTDIR</code> environment variable.	See GREOUTPUTDIR on page 858.

Overview	Reference
There is an improved architecture using HTTP for previewing documents in a distributed setup. Besides improvements in performance, the solution no longer requires the installation of a DVM on the remote machine.	See Distributed Mode on page 859.

Table 289: Genero BDL Reporting APIs

Overview	Reference
APIs support the Genero Report Viewer for HTML5:	See fgl_report_setBrowserDocumentDirectory on page 629, fgl_report_setBrowserDocumentDirectoryURL on page 629, fgl_report_setBrowserFontDirectory on page 630, fgl_report_setBrowserFontDirectoryURL on page 630
A new API supports distributed mode.	See fgl_report_configureDistributedURLPrefix on page 614.
APIs have been introduced to get error details.	See Functions to get error details on page 653.
A new API can programmatically set the value of environment or user-defined variables.	See fgl_report_setEnvironment on page 627.

Table 290: Genero Report Designer, Version 3.00

Overview	Reference
Genero Report Designer provides a LastPageFooter section property.	See section (Section) on page 754.
Support of Intelligent Mail bar code type.	See intelligent-mail on page 795.
New <code>smartParse</code> bar code property for bar code Code-128. When enabled, this allows you to enter the bar code value, and the internal code will be computed for you resulting in the shortest visual representation.	See smartParse (Smart Parse) on page 755 and code-128 on page 769.
New gs1* bar code aliases.	See Bar Code type listing on page 765.

Table 291: Graphical Debugger, Version 3.00

Overview	Reference
You can debug an already running process by attaching to the process. The process can be running locally or on a remote computer. Attaching to a remote process allows you to debug an application at a production site where Genero Studio is not installed.	See Start the Debugger on a running program on page 502.
You can debug an application deployed to a mobile device. With this new feature, the application is running on the mobile device and the Graphical Debugger is able to attach to the process.	See Debug a mobile application on page 504

Overview	Reference
You can debug Web services: server, client or both.	See Debug a Web services server application on page 504.

What's new in Genero Studio, v 2.51

This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication.

Table 292: General, Version 2.51

Overview	Reference
Genero Studio 2.51 includes features to support mobile app development for Android and iOS.	No additional reference.

What's new in Genero Studio, v 2.50

This publication includes information about new features and changes in existing functionality.

These changes and enhancements are relevant to this publication. See also [Migration notes for Version 2.50](#).

- [General](#)
- [Project Manager](#)
- [Form Designer](#)
- [Business Application Modeling](#)
- [Meta-schema Manager](#)
- [Graphical Debugger](#)
- [Source Code Management](#)
- [Genero Report Designer](#)
- [Genero Reporting APIs](#)

Table 293: General, Version 2.50

Overview	Reference
Mac OS support.	No further reference.
All build processes are done remotely for remote host configurations, optimizing speed and efficiency of remote builds on slower connections.	See Setting up a remote environment on page 154.
Configurations are stored on the remote host allowing users to reconnect from another machine with configurations intact.	See Migration notes for 2.50 upgrade guide on page 1023.
Improved file compatibility. If you open a file that had been saved with a prior version, you have the option to convert and open the file in the current version or to open in a different version of Genero Studio. When working in environments with multiple versions of Genero Studio installed, files will open in the appropriate version.	See Opening a file from a prior version on page 96.
Improved Diff tool. Modifications made to a file are shown as annotations, indicating added, modified and removed lines. You can now switch from a single pane view to a dual pane view showing the original file (as it is in the SVN repository for example) and the locally edited file.	See Using the Diff tool on page 380.

Overview	Reference
<code>-diff file1 file2</code> support. Genero Studio opens the given files in diff mode. This option is also integrated in File Browser.	See Command line options on page 103.
GAS Standalone (<code>httpdispatch</code>) can be automatically started by setting the startup script in the GAS configuration.	See Web: GAS/GWC configurations on page 147.
The Environment Variable definition dialog now includes a value list type.	See Add or edit environment variables on page 144
User documentation has been migrated to XML-based DITA (Darwin Information Typing Architecture).	No further reference.

Table 294: Project Manager

Overview	Reference
Project Manger extended to support other programming languages.	See Building and linking programs on page 343.
A Deploy Project <code>4pw</code> template (File >> New, Genero Files, Deploy Project) creates a directory structure recommended for a project which is to be deployed on the Genero Application Server (GAS).	No further reference.
Build Rules dialog and File Associations dialogs are accessible from the Tools menu (Tools >> Global setup or Tools >> Specific setup) and no longer in Preferences. The global setup overrides the built-in factory setup. It applies to all configuration of a user for the current machine or remote machine. The specific setup overrides the global setup and is specified by setting the <code>GSTSETUPDIR</code> variable in the Genero configuration.	See Migration notes for 2.50 upgrade guide on page 1023
User build rules, template build rules, user and template file associations are also stored on the server and not on the local machine allowing users to reconnect from another machine with build rules intact.	
<code>\$(tcl)</code> has been deprecated and replaced with <code>tclsh</code> in Business Application Modeling build rules. The <code>tclsh</code> executable generates the final file by using both a Tcl template file and the intermediary XML file crated by the <code>\$(generate)</code> command.	See tclsh on page 269.
Standard/error output redirection supported in build rules and user actions.	See Add/Edit a build rule on page 345.

Table 295: Form Designer

Overview	Reference
Graphical Topmenu editor allows for Topmenu editing directly in the form design.	See Add a Topmenu on page 436.
Improved ergonomics in the Form from Database wizard. Easier to create complex queries and select appropriate fields. Shows foreign key relations and NOT NULL constraints.	See Create form from database on page 409.
New and improved form item properties:	See Joins and Data order on page 453.

Overview	Reference
query	A single <i>query</i> property is now used to enter queries instead of several properties.
uniqueKey	Replaces <i>uniqueQueryKey</i> . Instead of having one <i>uniqueQueryKey</i> boolean on each field, one <i>uniqueKey</i> contains the list of unique fields.
Form properties benefit from a new initializer functionality which can set its value from another property.	See Form item properties on page 410.
Web components locations are specified with the environment variable <i>GSTWCDIR</i> and are no longer specified in Tools >> Preferences . Environment set configuration now includes a Web Components set.	See GSTWCDIR on page 144, Migration notes for 2.50 upgrade guide on page 1023.

Table 296: Business Application Modeling

Overview	Reference
New dbapp3.0 template with improved functionality:	See The default template features on page 199
<ul style="list-style-type: none"> • Properties in Business Application diagram allow for customization of the UI behavior. • Cascade delete supported in generated code. • Web Services can be generated and provide CRUD operations. • Improved architecture based on different layers: ui, webservice, uidata, database. • Improved insert, update, delete in the data layer. • Improved libdbapp architecture. • Dialog modularization used. Declarative <code>DIALOG</code> blocks are now defined as module elements and reused with the <code>SUBDIALOG</code> keyword of procedural <code>DIALOG</code> blocks. (See <i>DIALOG Block Structure</i> in the <i>BDL User Guide</i>.) • Centralized actions in the generated code. • The business record is checked to confirm that the unique key is defined as a primary key or a secondary key in the database schema. • The foreign key database constraint existence is checked. • Code generation uses unique constraint instead of unique index. • Constraints on the relations of Business Application diagram entities can be defined in the settings.agconf file. Validation is done on the constraints during editing. • Added native serial management for Informix® database. • Manages concurrent access. • New support for business record and form files without a database or a masterTable property set. • New relation properties allow for specifying the row position in an opened form and applying a SQL filter while opening a form. 	Form behavior in CRUD states on page 241 Cascade delete on page 232 Add Web services (Server, Services, Forms with services) on page 215 Add constraints or indexes on page 294 Managing SERIALs in a generated application on page 231 Managing concurrency on page 231 Add Relations on page 220

Overview	Reference
Business Application Modeling template files locations are specified with the environment variable <code>GSTSETUPDIR</code> and are no longer specified in Tools >> Preferences . Environment set configuration now includes a set for each available template set.	See GSTSETUPDIR on page 144, Migration notes for 2.50 upgrade guide on page 1023.
Create BA entities in the current view only instead of all views. New BA entities are only visible in the current view and are automatically hidden in the other views.	No further reference.
A new Toolbar has been added in the Business Application Modeler. This Toolbar contains all the items defined for the Business Application Modeler in the settings.agconf file.	No further reference.
Orphan properties now managed. Templates used in Application Generator define dynamic properties that can be used in objects such as records, form fields, and database objects. Opening a file generated with a template version different than the one set in Application Generator preferences may produce a warning indicating that some properties are not found in the current template definition. New warnings signal that those properties can now be removed using the Tools >> Specific setup >> Clean orphan properties menu option.	No further reference.
Added built-in Find .	No further reference.
Improved performance for large 4dbx database meta-schema files.	No further reference.

Table 297: Meta-schema Manager

Overview	Reference
Enhanced Revert action. A dialog is displayed listing changes made on the selected items in the schema with option to select which changes to revert.	See Revert schema changes dialog on page 315.
Database >> Diff Schema... option.	See Using the Diff tool on page 380.
Generate Database Update Script option. The Meta-schema Manager is able to generate a database update script. This script will modify the database according to the changes made in the Meta-schema. Previous version of the tables will be backed up and the data will be migrated to the new tables when applicable.	See Generate a database script from meta-schema on page 303.
Improved database creation script. The database creation script now supports Secondary Keys and case sensitivity and has improved support for Oracle MySQL and SQLite databases.	See Generate a database script from meta-schema on page 303.
Improved Locate in Diagram action.	See Meta-schema diagram context menu on page 318.
Create database objects in the current view only instead of all views. New Meta-schema objects are only visible in the current view and are automatically hidden in the other views.	No further reference.
Secondary Keys (also known as Unique Constraints) have been added in the Meta-schema Manager.	See Add constraints or indexes on page 294
Global meta-schema files are now specified with the environment variable <code>GSTSCHEMANAMES</code> and are no longer specified in Tools	See GSTSCHEMANAMES on page 143, Migration notes for

Overview	Reference
>> Preferences. Environment set configuration now includes a set for Global Database Schemas.	2.50 upgrade guide on page 1023.

Table 298: Graphical Debugger

Overview	Reference
Stack frames support, provided Genero 2.50 DVM is used. Navigate the call stack. Data view is automatically updated according to the current frame.	See Examining execution flow on page 509.
User variables of type ARRAY or RECORD are displayed in the Data view. Values can be re-assigned for complex types.	See Examining data on page 509.
Enhanced performance when retrieving and displaying data.	No further reference.
Support of signal <code>SIGINT</code> and signal <code>SIGQUIT</code> .	See Debugger (fgldb) command prompt on page 508, signal on page 525.

Table 299: Source Code Management (SCM)

Overview	Reference
SVN Blame command support and integration with Code Editor and SVN Lock command with a new view to manage all locks.	See Blame on page 537 and Locking on page 532.
Status view optimized for unversioned files.	See SVN Status view on page 543.
User and password are based on repository. Different users can now connect to different repositories at the same time.	No further reference.
Ignore List action now adds to the ignore list of the current directory only, not recursive up to the checkout directory.	No further reference.
SCM tasks are properly interrupted to avoid lock on the checkout.	No further reference.
Authentication is global to the checkout and is no longer folder based.	No further reference.

Table 300: Genero Report Designer

Overview	Reference
Tables support.	See Working with tables on page 691
Pivot tables support.	See Working with Pivot Tables on page 703.
Distributed mode. Allows the report engine to be started as a daemon to which Genero applications can connect to process the reports, allowing for vastly faster processing for short documents and improved scalability.	See Distributed Mode on page 859.
PDF enhancements. Improved PDF output, to include better memory consumption, use of the PDF referencing mechanism to improve Page M of N processing, share recurring images and CID keyed fonts support.	No further reference.
Null value support.	See The String Class on page 834 and The Numeric Class on page 821, Conditional Expressions .

Overview	Reference
Improved trigger updates. Algorithm improved to remove the need for frequent manual adjustments for each change within the data schema (rdd) file.	See Triggers on page 669.
Runtime localization. Report can now be localized independent of the language settings of the application.	See Change localization settings at runtime on page 598 and fgl_report_configureLocalization on page 616.
QR code barcode support.	See qr-code on page 796.
Display position of footers. Layout nodes designated as footers display at the bottom of the Mini Page, providing a WYSIWYG experience for the report designer.	See Page headers and footers on page 670.
Element creation by context. Create elements based on the document context in the report design. The object type created for a field is determined by the location in the document.	See Adding data values and captions on page 667.
Splitting of oversized elements across pages to prevent overflow.	See splitOversizedItem (Split Oversized Items) on page 755.
Rotation of items. The transformTransparently property changes the effect of the properties layoutDirection and swapX. When set, the transformation extends to the entire fragment so that entire documents can be rotated.	See transformTransparently (Transform transparently) on page 758.
Backside printing support.	See Backside printing on page 691.
Chart sorting. For MapCharts and CategoryCharts, the sortBy property allows you to specify how the data is sorted: alphabetic, numeric, or by order of declaration of the chart items. The sortAscending property allows you to sort in ascending or descending order.	See sortBy (Sort By) on page 755 and sortAscending (Sort Ascending) on page 755.
Fallback image support when the requested image for an Image Box is not found.	See Image Box on page 727.
Edit triggers with a Repeat selected items on menu option in the context menu in the Report Structure view, allowing you to select a trigger to be the parent of a document node.	See Place a trigger within the report structure on page 669.
Class property added for report elements.	See class (Class) on page 740.
Display and modify the sizing policy of containers.	See Modify the sizing policy of containers on page 664.
The fidelity property has been added to business charts and the pivot table, applied only when the object in question is drawn as a table.	See Business Graphs on page 731.
The layout direction of a parent container is highlighted in the Genero Report Designer by the addition of a dashed, slowly moving, U-shaped yellow border.	See layoutDirection (Layout Direction) on page 749.
Preference added to control the appearance of RTL expressions in the document view.	See Customize Report Designer: preferences on page 806.
Added options to facilitate the mass generation of images that are sized by their content (e.g. for web sites).	See fgl_report_setImageUsePageNamesAsFileNames on page 631 and

Overview	Reference
	fgl_report_setImageShrinkImagesToPageContent on page 631.

Table 301: Genero Reporting APIs

Overview	Reference
Specify the server where a Genero Report Engine is running in server mode.	See fgl_report_configureDistributedProcessing on page 614.
Configure the environment when the daemon is running on a different machine with different resource directories.	See fgl_report_configureDistributedEnvironment on page 613.
Distinguish between log entries originating from different users.	See fgl_report_setDistributedRequestingUserName on page 631.
“Postscript” has been added as an output format to the function <code>fgl_report_selectDevice</code> . The function <code>fgl_report_setPrinterWriteToFile</code> is deprecated.	See fgl_report_setPrinterWriteToFile on page 644 and fgl_report_selectDevice on page 626.
The Reporting API source has been split into two files, <code>libgre.4gl</code> and <code>libgreprivate.4gl</code> . These two files replace <code>helpers.4gl</code> .	See Reporting API Functions on page 599.
Function to control the paper orientation.	See fgl_report_setSVGOrientationRequested on page 646.
Switch off Unicode embedding. This is useful when the entire report uses Latin characters only, improving performance and document size.	See fgl_report_configurePDFFontEmbedding on page 620.
The function <code>fgl_report_configureAutoformatOutput</code> has changed from "sort by position of fields in PRINT statement" to "sort by matched pattern and then by position of field in PRINT statement."	See fgl_report_configureAutoformatOutput on page 612.
To follow the general pattern that calling a function with a NULL value exhibits the same behavior as not calling the function at all, the default values for the first three first parameters of the function <code>fgl_report_configureImageDevice</code> have changed to true.	See fgl_report_configureImageDevice on page 615.
The functions to load and commit report settings for the Genero Report Engine now search <code>FGLRESOURCEPATH</code> and <code>DBPATH</code> for relative file names. The function <code>fgl_report_findResourcePath</code> has been deprecated as a result.	See fgl_report_loadCurrentSettings on page 601, fgl_report_loadAndCommit on page 602, and fgl_report_findResourcePath on page 625.

What's new in Genero Studio, v 2.41

This topic lists features added for the 2.41 release of Genero Studio.

Table 302: Genero Studio, Version 2.41 New Features

Overview	Reference
Business Application Modeling (BAM) includes a new default Database Applications template, dbapp2.0.	See The code generation template set on page 942, Migrating to a new BAM template set on page 1027
The dbapp template in Genero Studio 2.40 was renamed in Genero Studio 2.41 to dbapp1.0.	See The code generation template set on page 942
dbapp2.0 includes new build.rules file for dedicating a set of build rules to a template.	See Build tab on page 359.
A new database schema file type, 4dbx, is used for generated applications.	See Database meta-schema (4dbx) on page 226

What's new in Genero Studio, v 2.40

This topic lists features added for the 2.40 release of Genero Studio.

Table 303: Genero Studio Version 2.40 New Features

Overview	Reference
Update settings in Source Code Manager have been updated. For example, you can now omit externals.	See Update / Update All on page 534.
You can now add multiple files.	See Add files on page 531.
The Meta-Schema Manager has been updated for easier editing and viewing. Options have been added in Edit mode, and document errors and warnings are displayed while editing.	See Meta-schema Manager on page 288.
Multiple views of the same database can be created.	See Viewing a meta-schema on page 300.
With Business Application Modeling, new business records have been added for reports.	See Add a Report Design Document (4rp) on page 213.
With Business Application Modeling, an improved editor assists with search and document errors.	No further reference.
With Business Application Modeling, new templates for a new architecture and for the generation or report code are provided.	See The code generation template set on page 942.
With Business Application Modeling, advanced customization is now possible.	See BAM Template Developer Guide on page 930.
With Business Application Modeling, an integrated code analyzer is provided.	See Code Analyzer on page 402.
With Report Writer, data matrix barcodes are supported.	See Bar Code type details on page 766.
With Report Writer, new properties for Text and Image fields allow them to behave as hyperlinks.	See Use hyperlinks in a report on page 689.

Overview	Reference
With Report Writer, additional options have been added for MapCharts.	See Map Chart on page 694.
With Report Writer, new report output format for Excel.	See Send report data to an Excel spreadsheet on page 594.
With Report Writer, new report output format for Word (rtf).	See Output report data in Microsoft RTF format on page 595 .
With Report Writer, there is a new function to create a process-level data file and a document at the same time	See fgl_report_setProcessLevelDataFile on page 644.
With Report Writer, there are new functions to create document metadata for compatibility reports	See Report Design Document metadata on page 673.
With Report Writer, there are new functions to provide generic report auto-formatting when no report design document (4rp) is specified for a report	See Auto-formatting Reports that have no 4rp (Report Design Document) on page 574.
With Report Writer, there are additional new report API functions, indicated by a "New in 2.40" designation.	See Reporting API Functions on page 599.
Additional support for application rendering with the Genero Web Client.	See Genero Configuration Management dialog on page 171.
You can now import configuration setup and preferences of earlier versions of Genero Studio when you first start up a new installation of Genero Studio.	See General Preferences on page 106.
For ease of use, the Window menu allows you to select the default layout, and you can toggle Full Size Documents or Full Screen to maximize your working space	No further reference.
States of previous sessions are remembered.	See Workspaces configuration on page 108.
Support of high contrast mode and of Windows™ screen readers for accessibility standards.	See Configuring Genero Studio on page 115
Multiple instances of Genero Studio can be opened simultaneously.	See Configuring Genero Studio on page 115
Various performance improvements have been implemented.	No further reference.

Upgrade Guides

Review the list of migration recommendations each time you move to a new version.

- [3.00 upgrade guide](#) on page 1023
- [2.50 upgrade guide](#) on page 1023
- [2.41 upgrade guide](#) on page 1025
- [2.40 upgrade guide](#) on page 1026
- [2.30 upgrade guide](#) on page 1026
- [2.20 upgrade guide](#) on page 1026

3.00 upgrade guide

Review when migrating to Genero Studio 3.00.

Genero Report Engine deployment

When in distributed mode, if the Genero Report Engine sits on a different machine than the runtime system (DVM) processing the report, you must do the following in order to preview reports:

- Start the Web server service (`grehttpd`).
- Set the `GREOUTPUTDIR` environment variable.

See [Distributed Mode](#) on page 859 for more details.

2.50 upgrade guide

Review when migrating to Genero Studio 2.50.

Add Remote Host Configurations

Configuration information is now kept on the host. This means that when a user connects to a remote host from a client, the host's configurations about the compiler, environment sets, GDC and GWC displays are available. To use your configurations from a prior version, you will need to follow these steps to re-associate the information to a named configuration. The named configurations are not transferred in 2.50, but the information to create them is available.

1. Start Genero Studio 2.50 on the client.
2. Select your remote host from the list in the lower right corner of Genero Studio. If you do not see your host in the list, follow these steps to add your remote host(s). See [Add a remote host](#) on page 155.
3. Select the wrench icon to display the [Genero Configuration Management dialog](#) on page 171. This dialog has changed to allow easy access to all configuration dialogs.

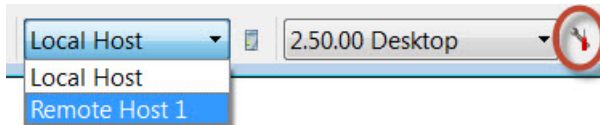


Figure 416: Select remote host and then configurations

4. Select the **Import Configuration** button. You will be prompted to choose a Genero Studio installation from which to import.

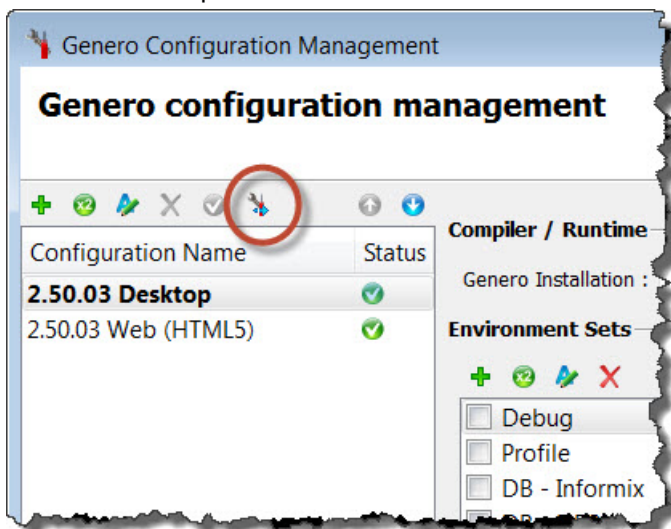


Figure 417: Import configurations

5. Select **Import**. Genero installations are imported and added to the **Genero Installation** list. Environment sets are imported and added to the **Environment Sets** list.
6. If you wish, add new **Configuration Names** to the list using the **Add** button. All of the configuration information is available on the remote host to use in your named configurations. The named configurations, however, are not imported and you may want to recreate a new named configuration in 2.50 for each prior version configuration that you had listed. Each configuration contains information about:
 - [Compiler / Runtime configuration \(Genero Installations\)](#) on page 172
 - [Environment sets](#) on page 140
 - [Desktop: GDC configurations](#) on page 146
 - [Web: GAS/GWC configurations](#) on page 147

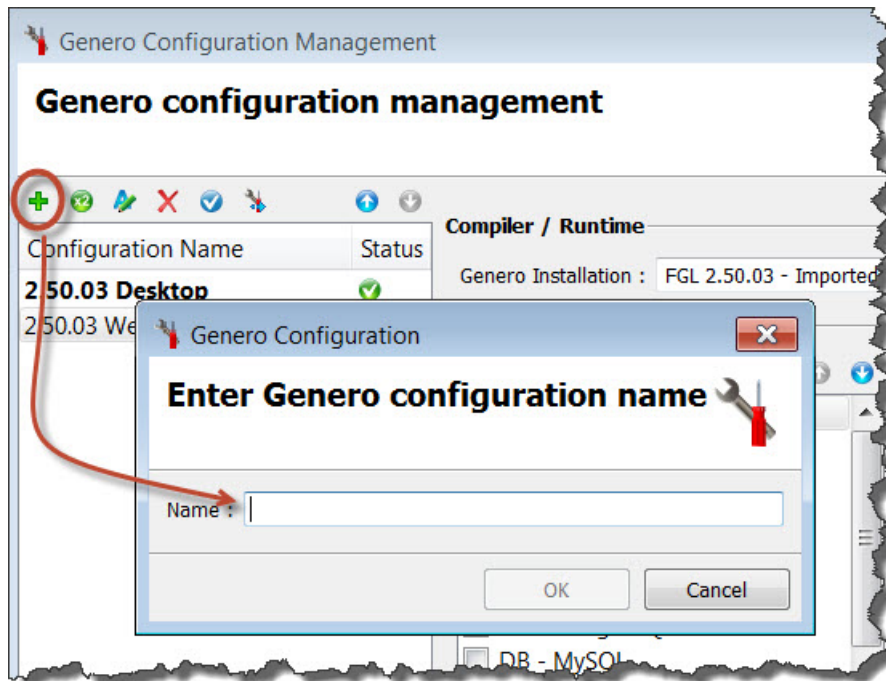


Figure 418: Add a named configuration

BAM Templates - GSTSETUPDIR

Business Application Modeling template files locations are specified with the environment variable *GSTSETUPDIR* and are no longer specified in **Tools >> Preferences**.

An environment set is listed for each default template set. Select the correct template set to be used.

If you use a custom template set, you will need to create an Environment Set and set *GSTSETUPDIR* on page 144 to the location of your template files. Be sure to check mark your environment set in the configuration(s) for which it is being used. See [Migrate customized template sets](#) on page 1029 for specific instructions.

Web Components - GSTWCDIR

Web components locations are specified with the environment variable *GSTWCDIR* and are no longer specified in **Tools >> Preferences**.

If you use web components, you will need to create an Environment Set and set *GSTWCDIR* on page 144 to the location of your web components. An environment set named **Web Components** is listed in the Environment Sets list and can be used as your web component environment set by setting its

GSTWCDIR to the location of your web components. Be sure to check mark your environment set in the configuration(s) for which it is being used.

Meta-Schemas - GSTSCHEMANAMES

It is recommended that you add schemas to projects so that they are loaded when the project is opened (and not at Genero Studio launch) and so that the project can be available to all developers without any additional configuration needed. However, specifying global schemas is still supported. Global meta-schema files are now specified with the environment variable *GSTSCHEMANAMES* and are no longer specified in **Tools >> Preferences**.

To make meta-schemas available to all projects and to appear in the DB Schemas tab, you can use the **Global Database Schemas** Environment Set and define the two environment variables within the environment set:

GSTSCHEMANAMES on page 143

Defines the filenames of the schemas to make available. (Do not include file extension.) Use the Value List environment variable type to list multiple meta-schemas separated by semi-colons.

FGLDBPATH

Defines the directories in which to find the schema files entered in the *GSTSCHEMANAMES* variable.

Be sure to check mark your environment set in the configuration(s) for which it is being used.

Genero Report Writer

The `fieldNamePatterns` input parameter for the reporting API function `fgl_report_configureAutoformatOutput()` has changed from "sort by position of fields in PRINT statement" to "sort by matched pattern and then by position of field in PRINT statement." If you have a report application that uses this function to define the output of an auto-format report that is not of a *COMPATIBILITY* type, verify that the report formats as expected and make modifications if necessary.

See [fgl_report_configureAutoformatOutput](#) on page 612.

Prior to version 2.50, localization information (`FGLPROFILE`, `FGLRESOURCEPATH`, `DBPATH`, `DBFORMAT` and so on) was statically defined at the start of a program and could not be changed at runtime. A work-around at the time was to place reports in separate executables, then to run these executables from the main application with a modified environment. This achieved the effect of modifying the localization configuration on a per report basis. This work-around will not work with reports run in distributed mode; the `fgl_report_configureLocalization` function should be used instead.

See [Change localization settings at runtime](#) on page 598.

2.41 upgrade guide

Review when migrating to Genero Studio 2.41.

Application Generator

- New default Database Applications template, `dbapp2.0`
- If you already have a BAM project, you should change the current template used to the environment set for the new template. See [Environment sets](#) on page 140.
- `dbapp2.0` includes new `build.rules` file for dedicating a set of build rules to a template.
- The `dbapp` template in Genero Studio 2.40 was renamed in Genero Studio 2.41 to `dbapp1.0`. The `dbapp` template set is the same template set as `dbapp1.0`.

2.40 upgrade guide

Version 2.40 migration notes.

The format of all Genero Studio XML files (4fd, 4rp, 4pw etc.) has been modified, and you will receive a warning if you execute an application compiled using Genero Studio versions 2.2x or 2.3x. For Genero Report Writer reports only, an error will occur if you have not converted the 4rp file. Opening a file in Genero Studio and saving it will convert it.

Form Designer

For Genero Studio form definition files (4fd), the gsform command-line tool can be used for a batch conversion of the files to version 2.40:

```
gsform -c <files>
```

Application Generation

The format of the settings.agconf file in the templates directory (<Studio-install-dir>/gst/bin/src/ag/tpl/dbapp/settings.agconf) has changed. If you have previously modified this file, you must make comparable changes in the 2.40 version of this file.

2.30 upgrade guide

Version 2.30 migration notes.

Report Writer

- TOP Margin is ignored in compatibility reports.

TOP MARGIN is now ignored when ASCII reports are run in compatibility mode. Margins can be set by using the API function fgl_report_setPageMargins().

- Parameter changed in fgl_report_configureCompatibilityOutput().

The **reportName** value for the fgl_report_configureCompatibilityOutput function is now automatically set to the name of the currently running report. When using the function, however, you must pass NULL as the value, for compatibility reasons.

- 4GL Boolean type is now interpreted as Numeric instead of String.

The 4GL **Boolean type** is now regarded as a Numeric type, instead of String as in previous versions. Existing expressions that contain references to boolean variables may require modification accordingly.

- START REPORT syntax must be TO XML HANDLER.

The BDL statement START REPORT <repname> TO XML HANDLER *handler* is now the only valid syntax for starting graphical reports using a 4rp design document.

2.20 upgrade guide

Version 2.20 migration notes.

Important: Genero Studio Form files (4fd), Project files (4pw), and Report files (4rp) are converted to the Genero Studio 2.20 format when opened. We strongly advise making a backup of your files before migrating them.

Meta-Schema Manager (formerly Database Browser)

1. Database Browser is replaced by Meta-schema Manager.
2. Schema availability has changed.

Form Designer

1. Form files (4fd) will be converted to the Genero Studio 2.20 form format when opened.
2. The Matrix container replaces the MFArray container.

Project Manager

1. Project files (4pw) will be converted to the Genero Studio 2.20 project format when opened.
2. The Genero variable FGLRESOURCEPATH defines the path to resource files.

Report Writer

1. Report files (4rp) will be converted to the Genero Studio 2.20 project format when opened.
2. The Reporting API is now linked as one library, `libgre.42x`.
3. The callback function `fgl_report_getFieldCaption()` is no longer called by default.
4. The Configuration menu has moved to File>>Report Properties.

Migrating to a new BAM template set

Topics to assist you with upgrading your generated application from one template set to another.

- [Migrate from dbapp3.2 to dbapp4.0](#) on page 1027
- [Migrate from dbapp3.1 to dbapp3.2](#) on page 1027
- [Migrate from dbapp3.0 to dbapp3.1](#) on page 1028
- [Migrate from dbapp2.0 to dbapp3.0](#) on page 1028
- [Migrate from dbapp1.0 to dbapp2.0](#) on page 1028
- [Migrate from 2.3x to dbapp1.0](#) on page 1029
- [Migrate customized template sets](#) on page 1029

Migrate from dbapp3.2 to dbapp4.0

To migrate your generated application from the dbapp3.2 template set to the dbapp4.0 template set, you simply need to confirm that you are using the dbapp4.0 template set.

The template dbapp4.0 provides the following benefits:

- The `DBAPP_MOBILE` environment variable can be set to 1 (TRUE) for generating applications for mobile devices. See [DBAPP_MOBILE](#) on page 268.
- In `settings.agconf`, the `editorInfo` attribute for MULTIPLELINES has changed for dialog titles to match other editorInfo syntax:
 - `bannerTitle` is now `description`.
 - `dialogTitle` is now `title`.

See [settings.agconf elements](#) on page 953 for details on the `settings.agconf`

Migrate from dbapp3.1 to dbapp3.2

To migrate your generated application from the dbapp3.1 template set to the dbapp3.2 template set, you simply need to confirm that you are using the dbapp3.2 template set.

The template dbapp3.2 provides the following benefits:

- The `canDisplay` and `canEmpty` functionalities can be specified for a form.
- Concurrent access management is disabled for generated mobile apps, as there is no need for a mobile app to manage concurrent access.

1. Select **Tools >> Genero Configurations**.

- In the [Environment sets](#) on page 140 list, select the environment set named **Template dbapp3.2**. This should be the only template environment set selected.

Migrate from dbapp3.0 to dbapp3.1

You can migrate your generated application from the dbapp3.0 template set to the dbapp3.1 template set.

- Confirm that you are using the dbapp3.1 template set. Select **Tools >> Genero Configurations**.
- In the [Environment sets](#) on page 140 list check mark the environment set named **Template dbapp3.1**.
- [Data refresh](#) on page 243 is a new feature in the dbapp3.1 template and affects relations between forms. After migrating templates, the default value of the property is **Current Row**, therefore if you want to have the same functional behavior as in dbapp3.0, you must set the `Data Refresh` property to **None** on all relations between forms.

Migrate from dbapp2.0 to dbapp3.0

You can migrate your generated application from the dbapp2.0 template set to the dbapp3.0 template set.

- Confirm that you are using the dbapp3.0 template set. Select **Tools >> Genero Configurations**.
- In the [Environment sets](#) on page 140 list check mark the environment set named **Template dbapp3.0**.
- Confirm that your business records each have a unique key. Set the unique key if it has not already been defined as a primary key or secondary key in the database meta-schema.
- The `Source Field` and `Destination Field` properties have been replaced by the `Source Field` in the group **Position** and by `Source Field/Destination Field` in the group **Filter**. Consequently, you may need to update some relations in your Business Application Diagram. When you open your existing BA diagram, the values of the orphan property `Source Field` are in the `Source Field` of the group **Position** and the `Destination Field` property becomes an orphan property. Update your relations between forms:

Option

Description

Relation between a Form and a Form

- In the **Filter** group, update the `Source Field` property with the same values as those in the `Source Field` property in the group **Position**.
- Update the `Destination Field` with the same values as those in the `Destination Field`.
- In the **Position** group, reset the values specified in the property `Source Field`.

Relation between a Form and a Zoom

No update needed.

- The dbapp3.0 template set is a new application architecture thus some BLOCK/POINT sections are in different source files. To migrate your modified BLOCK/POINT sections, please contact your support center.

Migrate from dbapp1.0 to dbapp2.0

You can migrate your generated application from the dbapp1.0 template set to the dbapp2.0 template set.

- Modify your `4pw` project file by removing any Application Generator file build rules. Delete the duplicate rules from the **Edit Build Rules** dialog so that your program will use the new template build rules.
- The dbapp2.0 build rules are in a file called `build.rules`. If you have your own template directory, you must copy the `build.rules` file from the dbapp2.0 template directory. If you have modified the build rules, reintegrate your changes in the `build.rules` file.
- The dbapp2.0 template uses the `4dbx` database schema file type. Before using the dbapp2.0 template, replace the `4db` files with `4dbx` ones.

4. Open the 4db file and use **Save As** to save the file with a 4dbx extension. If you have defined your own template directory:
 - a) Confirm that you have the 4dbx file type definition.
 - b) Confirm that the actions are present in the `creatable.conf`.
 - c) Remove the build rules from your projects if you have added them (or update them to execute the same operations as the provided ones).
 - d) Confirm that the `build.rules` file contains the same build rules as the provided one for 4dbx.
 - e) Confirm that the `settings.agconf` contains the 4dbx section.
5. Report instruction names have changed. For each 4rp file, you need to select its data source again. The data source can be created by building your 4rd file.
6. Confirm that you are using the dbapp2.0 template set. Select **Tools >> Preferences, Application Generator** and select **Database Applications 2.0**.

Migrate from 2.3x to dbapp1.0

You can migrate your generated application from a 2.3x template set to the dbapp1.0 template set.

This procedure assumes you have not modified your source files with code in BLOCK and POINT sections or modified the Tcl files. If so, you will have to re-generate the program and then manually add your code changes to the newly generated code.

1. Open the project to be migrated from 2.3x to dbapp1.0.
2. Right-click on the 4pw and select **Edit Build Rules**. Remove any user-defined build rules.
3. Save and close your project.
4. Select **Tools >> Genero Configurations** and select the dbapp 1.0 environment set. This will now be the code generation template used for the project.
5. Open your project.
6. Remove `libag.42m` from the **External Dependencies** of the **Library** node and save the project.
7. The CRUD Form and Zoom Form types replace the Module type. Open the Business Application Diagram. For every Module entity corresponding to a form, right-click on the entity and select **Convert to >> CRUD Form**. For every Module entity corresponding to a zoom, right-click on the entity and select **Convert to >> Zoom Form**.
8. Save the project.
9. Rebuild the project.

Migrate customized template sets

You can migrate your customized template set.

1. Confirm that you are using your template set. Select **Tools >> Genero Configurations**.
2. In the [Environment sets](#) on page 140 list check mark the environment set for your template, or create one pointing [GSTSETUPDIR](#) on page 144 to your template set.
3. Open the `settings.agconf` file and modify the version attribute of the root element to 5.


```
<AGSettings version="5">
```
4. Open the other settings files in the Code Editor and check if there are errors (file-types.xml, `creatable.conf`, and `build.rules`).
5. In case of errors, review the template setting files to which you want to migrate and modify your settings to match the new file format.
6. A new template may generate different code. If you have modified the template files, you may lose those changes when using a new template set. If you want to use a new template, you may have to manually integrate your changes into the new template set.

If the Application Generator model is identical or there is no impact on your template files or you are not interested in the new template features, you can keep your templates unchanged. If you have modified the generated source (4gl) code in a BLOCK or a POINT, changing the current template does

not delete your changes. If the corresponding BLOCK/POINT still exists in the template, your changes will be integrated during the next compilation. If the BLOCK/POINT has been removed from the template, it will appear at the end of the file in a lost BLOCK/POINT, that you can reintegrate into your code (with copy/paste, for example).

Legal Notices

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software developed by CollabNet (<http://www.Collab.Net/>).

This product includes software developed by the University of California, Berkeley and its contributors.

This product includes software developed or owned by Caldera International, Inc