# 1 Aubit4GL Quick Reference

Version 0.4: 5 Oct 2006

## 1.1 Data Types

```
ARRAY[m,n,...] OF type
BYTE
CHAR(n)
CHARACTER(n)
DATE
DATETIME(f TO l)
DEC
DEC(m)
DEC(m,n)
DECIMAL
DECIMAL(m)
DECIMAL(m,n)
DOUBLE PRECISION
DOUBLE PRECISION(n)
INT
INTEGER
INTERVAL(f TO l)
LIKE table.column
MONEY
MONEY(m)
MONEY(m,n)
NUMERIC
NUMERIC(m)
NUMERIC(m,n)
REAL
RECORD LIKE table.*
RECORD name type ,... END RECORD
SERIAL
SERIAL(n)
SMALLFLOAT
SMALLINT
TEXT
VARCHAR
VARCHAR(m)
VARCHAR(m,n)
```

Current Engines also support large integers: int8 and serial8.

## 1.2 Constants

```
TRUE=1
FALSE=0
NOTFOUND=100
```

## 1.3 Global Variables

**Flags:** INT_FLAG QUIT_FLAG

**Vars:** STATUS SQLCA.SQLCODE

**SQLCA Record:**

```
SQLCA RECORD
SQLCODE INTEGER,
SQLERRM CHAR(71),
SQLERRP CHAR(8),
SQLERRD ARRAY[6] OF INTEGER,
SQLAWARN CHAR(8)
END RECORD
```

**SQLCA.SQLERRD Array:**

```
SQLERRD[1]:estimated row count
SQLERRD[2]:serial value returned
SQLERRD[3]:no of rows processed
SQLERRD[4]:estimated CPU cost
SQLERRD[5]:error offset
SQLERRD[6]:last rowid processed
```

Warning: Not all of the above work for all backends. For PostgreSQL they may need a patched version of the engine.

## 1.4 Syntax Conventions

The remainder of this chapter uses the following conventions to indicate the syntax of 4GL language constructs

- KEYWORDS are in UPPERCASE. You enter them literally but in upper or lower case
- Lower case indicates terms for which you must enter your own identifiers or expressions
- "string" indicates a quoted string. Informix allows either single or double quotes but non-Informix engines may enforce one or the other.
- string (without quotes) indicates an unquoted string used for example, in naming cursors, prepared statements, forms, windows, etc.
- m and n are used to denote a numeric value
- "c" denotes any quoted character
- [] and {} delimit options. {} indicates a mandatory option. [] a non-mandatory toption. Within the [] or {} elements are separated by the pipe symbol |. e.g. {a|b|c} means you must choose a or b or c.
- Expressions in red are Aubit 4GL extensions and will not compile on Informix, 4J, or other 4GL compilers.
- relop means a relational operator (see below)
- expr means an expression
- charexpr means a character expression (e.g. `filename || ".4gl"`)

## 1.5 Operators

**Numeric:** + - * / ** mod

**String:** , [m,n] || USING CLIPPED

**Relational:** = <> != >= < <=

**Boolean:** expr relop expr

```
charexp LIKE charexpr
charexpr LIKE charexpr ESCAPE "c"
charexp NOT LIKE charexp
charexp NOT LIKE charexp ESCAPE "c"
charexpr MATCHES charexpr
charexpr NOT MATCHES charexpr ESCAPE "c"
charexpr MATCHES charexpr
charexpr NOT MATCHES charexpr ESCAPE "c"
expr IS NULL
expr IS NOT NULL
boolexpr AND boolexpr]
boolexpr OR boolexpr
NOT boolexpr
```

## 1.6 Attribute Constants

An attlist is a set of the following elements:

BLACK, WHITE, RED, GREEN, BLUE,
MAGENTA, CYAN, YELLOW,
REVERSE,DIM, BOLD, BLINK, INVISIBLE,
BORDER, UNDERLINE

## 1.7 Key Constants

A keylist is a set of the following elements:

```
F1 to F64
CONTROL-c (but c not in (A,D,H,I,
  K,L,M,R,X)
ACCEPT, DELETE, DOWN, ESC, ESCAPE,
HELP, INSERT, INTERRUPT, LEFT,
RIGHT, NEXT, NEXTPAGE, PREVIOUS,
PREVPAGE, RETURN, TAB, UP
```

## 1.8 Table Privileges

```
ALTER, INDEX, DELETE, INSERT,
SELECT[(colname ,...)]
UPDATE[(colname ,...)
```

## 1.9 4GL Statement Syntax

```
ALLOCATE ARRAY
ALTER INDEX indexname TO [NOT] CLUSTER
ALTER TABLE tablename
  {ADD (newcolname newcoltype
     [BEFORE old-colname][,...])
   |DROP (oldcolname[,...])
   |MODIFY (oldcolname newcoltype [NOT NULL]
     [,...])
  }[,...]
AT TERMINATION CALL function([args])
BEGIN WORK
  statement ...
  {COMMIT WORK | ROLLBACK WORK}
CALL [packet{::|.}]function([args])
  [RETURNING arglist]
CASE [(expr)]
  WHEN {expr | booleanexpr}
    statement
    ...
    [EXIT CASE]
  ...
  [OTHERWISE]
    ...
    [EXIT CASE]
    ...
END CASE
CHECK MENUITEM
CHECK MENUITEM2
CLEAR STATUSBOX
CLEAR {SCREEN |WINDOW windowname
   |WINDOW SCREEN
   |FORM [TO DEFAULTS]
   |fieldlist}
CLOSE cursor
CLOSE DATABASE
CLOSE FORM
CLOSE SESSION
CLOSE STATUSBOX
CLOSE WINDOW name
CODE
  Cstatement;
  ...
  ENDCODE
COMMIT WORK
CONNECT
CONSTRUCT {BY NAME charvar ON collist
  |charvar on collist FROM { fields
     | screenrecord[[n]].*}  [,...]}
  [[ {BEFORE|AFTER} CONSTRUCT statements]
   [,...]
   [ {BEFORE|AFTER} FIELD field statements]
   [,...]
   END CONSTRUCT]
CONTINUE CONSTRUCT
CONTINUE DISPLAY
CONTINUE FOR
CONTINUE FOREACH
CONTINUE INPUT
CONTINUE MENU
CONTINUE PROMPT
CONTINUE WHILE
CONVERT REPORT TO "filename" AS
  {"SAVE"|"PDF"|"CSV"|"TXT"}
CREATE AUDIT FOR tabname in "pathname"
CREATE [UNIQUE|DISTINCT][CLUSTER] INDEX
 indname ON tabname( colname [ASC|DESC]
       [,...])
CREATE DATABASE {name| charvar}
   [WITH LOG [IN path]]
CREATE SCHEMA AUTHORIZATION
CREATE PRIVATE SYNONYM
CREATE PUBLIC SYNONYM
CREATE SYNONYM name FOR tabname
CREATE TABLE
CREATE [TEMP] TABLE name
   (colname coltype [NOT NULL][,...])
CREATE DISTINCT CLUSTER INDEX
CREATE VIEW
CURRENT WINDOW IS name
CURRENT WINDOW SCREEN
CURRENT WINDOW IS SCREEN
DATABASE name [EXCLUSIVE]
DEALLOCATE ARRAY name
DECLARE name [SCROLL] CURSOR FOR
  {select_statement
    [FOR UPDATE OF collist
  |insert_statement
  |statementid}
DEFER INTERRUPT
DEFER QUIT
DEFINE varlist datatype [,...]
DEFINE CONSTANT id {"string"|number}
DEFINE name ASSOCIATE [CHAR](n)
   with ARRAY[m] OF datatype
DELETE FROM tabname
   [WHERE {condition|CURRENT OF cursor}]
DELETE USING
DISABLE
DISABLE FORM
DISABLE MENUITEM
DISABLE MENUITEMS
DISPLAY {BY NAME varlist
  | varlist TO {fieldlist|screenrec[[n]].*}
      [,...]
    | AT screenrow,screencol]}
DISPLAY ARRAY id TO screenarray.*
   [ ATTRIBUTE( attlist )]
   {ON KEY (keylist)
      statement
      ...
      [EXIT DISPLAY]
   ...
   END DISPLAY| [END DISPLAY]}
```

```
DISPLAY FORM name [ATTRIBUTE( attlist)]
DROP AUDIT FOR tabname
DROP DATABASE {name | charvar}
DROP INDEX name
DROP SYNONYM name
DROP TABLE name
DROP TRIGGER name
DROP VIEW name
ENABLE
ENABLE FORM
ENABLE MENUITEM
ENABLE MENUITEMS
ERROR displaylist [ATTRIBUTE (attlist)]
EXECUTE [IMMEDIATE] statementid
EXIT CASE
EXIT CONSTRUCT
EXIT DISPLAY
EXIT FOR
EXIT FOREACH
EXIT INPUT
EXIT MENU
EXIT PROGRAM [expr]
EXIT PROMPT
EXIT WHILE
FETCH [NEXT
      |PREVIOUS|PRIOR|FIRST|LAST
        |CURRENT|RELATIVE n
        |ABSOLUTE n]
     cursorname [INTO varlist]
FINISH REPORT name
   CONVERTING TO "filename"|EMAIL|[AS
"type"]|MANY
   [USING "filename" AS LAYOUT]
FLUSH cursor
FONT SIZE n
FOR var = expr TO expr [STEP expr]
   {statement|CONTINUE FOR|EXIT FOR}...
   END FOR
FOREACH cursor [INTO varlist]
   [statement|CONTINUE FOREACH|EXIT FOREACH]...
   END FOREACH
FREE {statementid|cursor|blobvar}
FREE REPORT name
FUNCTION function([arglist])
   [ definestatement ]...
   statement ...
   END FUNCTION
GO TO label
GOTO label
GRANT {tabpriv ON tabname
   | CONNECT|RESOURCE|DBA }
   TO {PUBLIC|userlist}
HIDE OPTION name
HIDE WINDOW name
IF boolexpr THEN
  statement
  ...
  [ELSE
    statement
    ...
  END IF]
IMPORT PACKAGE name
INITIALIZE varlist
  {LIKE collist| TO NULL}
INPUT ARRAY array [WITHOUT DEFAULTS]
  FROM screenarray.* [HELP n]
  [{BEFORE {ROW|INSERT|DELETE|FIELD list}
    [,...]
    |AFTER {ROW|INSERT|DELETE|FIELD list
       INPUT}[,...]
     |ON KEY (keylist)}
     statement
     ...
     [NEXT FIELD field]
     ...
     [EXIT INPUT]
     ...
   ...
   END INPUT
INSERT INTO tabname[(collist)]
  {VALUES(vallist)| selectstatemet}
LABEL name:
MESSAGE displaylist [ATTRIBUTE (attlist)]
LABEL label-name :
LET id = expr
LET hasharray<<"code">> = "string"
LOAD FROM filename INSERT in tabname [(collist)]
LOCATE varlist in {MEMORY|FILE [filename]}
LOCK TABLE name IN {SHARE|EXCLUSIVE} MODE
MENU "name"
   COMMAND {KEY (keylist)
       | [KEY (keylist)] "option"
           [HELP n]}
      statement
      ...
      [CONTINUE MENU]
      ...
      [EXIT MENU]
      ...
      [NEXT OPTION "option"]
      ...
    ...
    END MENU
MESSAGEBOX message
SET BUFFERED LOG
SET CONSTRAINTS ALL IMMEDIATE
SET LOG
START DATABASE identifier WITH LOG IN "..."
  [MODE ANSI]
START REPORT name
   [TO {file|PIPE program|PRINTER|CONVERTABLE}
MOVE WINDOW
NEED n LINES
NEXT FIELD "fieldname"
NEXT FORM
NEXT OPTION "optname"
OPEN cursor [USING varlist]
OPEN FORM name FROM "filename"
OPEN SESSION
OPEN STATUSBOX
OPEN WINDOW name AT row, col
   WITH {r ROWS, c COLUMNS
         | FORM "file"}
   [ATTRIBUTE(attlist)]
OPTIONS {MESSAGE LINE line
   |PROMPT LINE line
   |COMMENT LINE line
   |ERROR LINE line
   |FORM LINE line
   |INPUT {[NO] WRAP}
   |INSERT KEY key
   |DELETE KEY key
   |NEXT KEY key
   |PREVIOUS KEY key
   |ACCEPT KEY key
   |HELP FILE "file"
   |HELP KEY key
   |INPUT ATTRIBUTE(attlist)
   |DISPLAY ATTRIBUTE (attlist)}
   [,...]
OUTPUT TO REPORT name(exprlist)
PAUSE
PREPARE
PRINT
PRINT FILE
PRINT IMAGE
PROMPT displaylist FOR [CHAR] var
  [HELP n]
```

```
  [ON KEY (keylist)
    statement
    ...
  ...
  END PROMPT]
PUT cursor FROM varlist
RECOVER TABLE name
RENAME COLUMN table.oldcol TO newcol
RENAME TABLE oldname TO newname
RESIZE ARRAY
EXIT REPORT
RETURN  exprlist ]
REVOKE { tabpriv ON tabname
   | CONNECT | RESOURCE | DBA}
   FROM {PUBLIC | userlist
ROLLBACK WORK
ROLLFORWARD DATABASE name
RUN command [RETURNING n
   |WITHOUT WAITING]
SCROLL {fieldlist| screenrec.*}[,...]
   {UP|DOWN}[BY n]
SELECT
SELECT USING
SET PAUSE MODE OFF
SET PAUSE MODE ON
SET CURSOR
SET SESSION
SET SESSION TO
SHOW MENU
SHOW OPTION "optname"
SHOW WINDOW name
SKIP n LINE[S]
SKIP BY nval
SKIP TO nval
SKIP TO TOP OF PAGE
SLEEP n
SQL sqlstatement [,...] END SQL
SET EXPLAIN OFF
SET EXPLAIN ON
SET ISOLATION TO COMMITTED READ
SET ISOLATION TO CURSOR STABILITY
SET ISOLATION TO DIRTY READ
SET ISOLATION TO REPEATABLE READ
SET LOCK MODE TO NOT WAIT
SET LOCK MODE TO WAIT
START EXTERNAL FUNCTION
START REPORT name
   [TO {"filename"|PIPE program|PRINTER}]
STOP ALL EXTERNAL
TERMINATE REPORT
UNCHECK MENUITEM
UNCHECK MENUITEMS
UNLOAD TO filename selectstatement
UNLOCK TABLE name
UPDATE tabname SET
   {colname = expr [,...]
   |{(collist}|table.*|*}=
      {(exprlist)| record.*}}
   [WHERE {condition|CURRENT of cursor}
UPDATE STATISTICS
UPDATE STATISTICS FOR TABLE name
UPDATE USING
USE packagename
USE SESSION name
VALIDATE var LIKE collist
WHILE boolean
   [statement| EXIT WHILE | CONTINUE WHILE}...
   END WHILE
```

## 1.10 Report Syntax

```
REPORT repname( arglist)
  definestatement,...
[OUTPUT
  [REPORT TO
    {file|PIPE program|PRINTER}]
  [LEFT MARGIN n]
  [RIGHT MARGIN n]
  [TOP MARGIN n]
  [BOTTOM MARGIN n]
  [PAGE LENGTH n]
[ORDER [EXTERNAL] BY sortlist]
FORMAT
  { EVERY ROW
   | {[FIRST] PAGE HEADER
      |PAGE TRAILER
      |ON EVERY ROW
      |ON LAST ROW
      |{BEFORE|AFTER} GROUP OF argvar}
       statement
       ...
       [...]}
END REPORT
```

## 1.11 Report Statement Syntax

```
NEED n LINES
PAUSE "string"
PRINT [[exprlist][;]| FILE "filename"]
SKIP {expr LINE[S]| TO TOP OF PAGE}
```

## 1.12 Report Expressions

```
COLUMN expr
[GROUP]{COUNT(*)|PERCENT(*)
  |{SUM|AVG|MIN|MAX}(expr)}
   [WHERE expr]}
DATE
LINENO
PAGENO
TIME
WORDWRAP
```

## 1.13 PDF Report Statements

```
NEED n LINES
PAUSE "string"
PRINT [[exprlist][;]| FILE "filename"]
PRINT IMAGE blobvar AS
   "{GIF|PNG|TIFF|JPEG}"
   [SCALED by x.n, y.n}
SKIP {expr LINE[S]| TO TOP OF PAGE}
SKIP {BY|TO} nval
CALL PDF_FUNCTION(arglist)
```

## 1.14 PDF Report Syntax

PDF reports are an Aubit 4GL extension.

- nval means an numeric expr followed by 1 of the following units:
  POINTS, INCHES, MM, or nothing (which means char spaces). Example: 2.54 mm

```
PDFREPORT name(arglist)
  definestatement ,...
[OUTPUT
  [REPORT TO
    {file|PIPE program|PRINTER}]
  [LEFT MARGIN nval]
```

```
  [RIGHT MARGIN nval]
  [TOP MARGIN nval]
  [BOTTOM MARGIN nval]
  [PAGE LENGTH nval]
[ORDER [EXTERNAL] BY sortlist]
FORMAT
  { EVERY ROW
   | {[FIRST] PAGE HEADER
      |PAGE TRAILER
      |ON EVERY ROW
      |ON LAST ROW
      |{BEFORE|AFTER} GROUP OF argvar}
        statement| pdfstatement
        ...
        [...]}
END PDFREPORT
```

## 1.15 PDF Report Expressions

```
COLUMN nval
reportexpression
```

# 2 Builtin Functions

Informix 4GL has a set of 40 or more functions built in to the language. Aubit4GL implements all of these.

Aubit4gl also implements a few functions to make the compiler compatible with programs written for D4GL.

Finally Aubit4GL has added some builtins of its own to allow you to exploit Aubit4GL's special features such as GUI interfaces, different database engines, etc.

## 2.1 Standard 4GL Builtin Functions

The following functions in 4GL work in Aubit4GL:

| Function | Comment |
|---|---|
| arg_val(n) | |
| arr_count() | returns smallint |
| arr_curr() | returns smallint |
| downshift(s) | returns a string |
| err_get(n) | returns a string |
| err_print(n) | displays a string |
| err_quit(n) | displays a string then exits |
| errorlog(s) | logs message s to logfile |
| fgl_drawbox(h,w, y, x [,colour]) | |
| fgl_getenv(s) | returns string |
| fgl_keyval(s) | returns integer code |
| fgl_lastkey() | returns integer code |
| length(s) | returns smallint |
| num_args() | returns smallint |
| scr_line() | returns smallint |
| set_count(n) | |
| showhelp(n) | displays help message n |
| sqlexit(n) | returns 0, after closing connection to database |
| startlog(s) | |
| upshift(s) | returns string with chars downshifted to lowercase |

## 2.2 Standard 4GL Operators

The following functions are described by Informix 4GL as builtin operators. They work in Aubit4GL:

| Operator | Comment |
|---|---|
| ascii(n) | returns a char, e.g. ascii(64) returns 'A' |
| date(s) | returns a date |
| date | returns a string e.g. Wed Aug 15 2006 |
| day(d) | returns 1..31 |
| extend(d or dt, format) | returns a date or datetime |
| field_touched(rec.field) | returns TRUE or FALSE |
| get_fldbuf(rec.field) | returns string contents of field |
| hex(n) | returns string e.g. 0x0000001c |
| in() | |
| infield(rec.field) | returns TRUE or FALSE |
| mdy(m,d,y) | returns date from args month, day, year |
| month(d or dt) | returns 1:12 |
| ord(c) | returns smallint |
| time | returns string |
| today | returns date |
| year(date) | returns smallint |

## 2.3 D4GL Builtin Functions

The following are do-nothing functions which allow 4J's D4GL programs to compile:

| Function | Comment |
|---|---|
| ddeconnect() | |
| ddeexecute() | |
| ddefinish() | |
| ddefinishall() | |
| ddegeterror() | |
| ddepeek() | |
| ddepoke() | |
| | |

## 2.4 Aubit Builtin Functions

First cut here. Give me time to study how to document them!

| Function | Comment |
|---|---|
| abs( n ) | returns absolute value of n |
| a4gl_get_info("t","id","p") | See separate outline |
| a4gl_get_page() | |
| a4gl_get_ui_mode() | |
| a4gl_run_gui() | |
| a4gl_set_page() | |
| a4gl_show_help() | |
| dbms_dialect() | |
| fgl_buffertouched() | |
| fgl_dialog_get_buffer() | |
| fgl_dialog_getfieldname() | |
| fgl_dialog_setbuffer() | |
| fgl_dialog_setcurrline() | |
| fgl_dialog_setkeylabel() | |
| fgl_getkey_wait() | |
| fgl_setkeylabel(s) | |
| fgl_prtscr() | |
| fgl_scr_size() | |
| fgl_set_arr_line() | |
| fgl_keysetlabel() | |
| fgl_set_scrline() | |
| fgl_strtosend() | |
| fgl_winmessage() | |
| load_datatype() | |
| set_window_title(s) | |
| sqrt(n) | returns square root of n |
| winexec(s) | |
| winexecwait() | |

## 2.5 a4gl_get_info()

**Synopsis:** a4gl_get_info( "object", "id", "property")
where "object" in ("Form"|"Window"|"Connection"|"Statement")
and "id" is the quoted variable name of instance of the object
and property is an element of the set of properties of the object as follows:

In the properties below, replace the % with a value 1 .. maxelement.

### 2.5.1 Connection

**Synopsis:** a4gl_get_info("Connection", "", "Database")

Database in the only property available. The id argument is ignored.

### 2.5.2 Form

| Form Property | Return Value |
|---|---|
| Database | char |
| Delimiters | char |
| ScreenRecordCount | int |
| ScreenRecordName% | int |
| AttributeCount | int |
| CurrentField | int |
| Width | int |
| Height | int |
| Field% | long? |
| ScreenName% | char |
| TableName% | char |
| AliasName% | char |
| FieldType% | char |
| FieldSize | int |
| FieldBytes | int |
| FieldDets | long |
| Screens | long |

### 2.5.3 Statement

| Atatement Property | Return Value |
|---|---|
| NoColumns | int |
| NoRows | int |
| Name% | char |
| Type% | char |
| Scale% | int |
| Nullable% | int |
| Length% | int |

### 2.5.4 Window

| Window Property | Return Value |
|---|---|
| Height | int |
| Width | int |
| BeginX | char |
| BeginY | char |
| Border | int |
| Metrics | int,int,int,int (x, y, h, w) |