

RSA SecurID®
Software Token Converter 3.1
Administrator's Guide



Contact Information

See the RSA corporate web site for regional Customer Support telephone and fax numbers: www.emc.com/domains/rsa/index.htm

Trademarks

RSA, the RSA Logo, EMC, and SecurID are either registered trademarks or trademarks of EMC Corporation in the United States and/or other countries. All other trademarks used herein are the property of their respective owners. For the most up-to-date listing of RSA trademarks, go to www.emc.com/legal/emc-corporation-trademarks.htm#rsa.

License agreement

This software and the associated documentation are proprietary and confidential to EMC, are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice below. This software and the documentation, and any copies thereof, may not be provided or otherwise made available to any other person.

No title to or ownership of the software or documentation or any intellectual property rights thereto is hereby transferred. Any unauthorized use or reproduction of this software and the documentation may be subject to civil and/or criminal liability. This software is subject to change without notice and should not be construed as a commitment by EMC.

Note on encryption technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

Distribution

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Contents

Preface	5
About This Guide.....	5
RSA SecurID Software Token Converter Documentation	5
Related Documentation.....	5
Support and Service	6
Before You Call Customer Support.....	6
Chapter 1: Overview	7
Introduction.....	7
What's New in Token Converter 3.1	8
Backwards Compatibility	8
JRE Requirement	8
Supported Mobile Apps	9
Supported Operating System Platforms.....	9
Supported Provisioning Servers.....	9
Chapter 2: Using RSA SecurID Software Token Converter	11
Before You Begin	11
Token Converter Input and Output	12
Running the Token Converter.....	12
Run the Token Converter on Windows	13
Run the Token Converter on UNIX.....	13
Prerequisite for Using Red Hat Linux	13
Command Line Usage.....	14
Platform Target Options	14
Token Options.....	15
CTF Output Formatting Options.....	15
File Output Options	16
- version	16
Examples	17
Token Converter Version.....	17
iOS 2.0 App	17
iOS 2.0 SDK	18
iOS (iPhone and iPad) 1.3 App.....	19
iOS (iPhone and iPad) 1.5 SDK.....	19
Android 2.0 App	20
Android 2.0 SDK	21
Android 1.2 App	22
Android 1.2 SDK	22
Windows Phone 1.0 App	23
Chapter 3: Error Messages	25
Error Messages.....	25

Preface

About This Guide

This guide describes how to use the RSA SecurID® Software Token Converter 3.1 (Token Converter) command line utility to convert an RSA SecurID software token into a custom compressed token format (CTF) URL or a QR Code. This guide is intended for administrators who have experience provisioning software tokens to RSA SecurID mobile apps.

RSA SecurID Software Token Converter Documentation

Administrator's Guide (this guide). Describes how to use the Token Converter to convert software token files for easy delivery to supported mobile device platforms.

Release Notes. Describes changes in this release, known issues and workarounds, and other information pertinent to the release.

Related Documentation

RSA SecurID Software Token for iOS Administrator's Guide (versions 2.0 and 1.3). Describes how to provision and deliver software tokens to iOS devices running the RSA SecurID software token app.

RSA SecurID SDK for iOS Developer's Guide (versions 2.0 and 1.5). Describes how to use the SDK to integrate RSA SecurID one-time password (OTP) features directly into a third-party iOS app.

RSA SecurID Software Token for Android Administrator's Guide (versions 2.0 and 1.2). Describes how to provision and deliver software tokens to Android devices running the RSA SecurID software token app.

RSA SecurID SDK for Android Developer's Guide (versions 2.0 and 1.2). Describes how to use the SDK to integrate RSA SecurID one-time password (OTP) features directly into a third-party Android app.

RSA SecurID Software Token 1.0 for Windows Phone Administrator's Guide. Describes how to provision and deliver software tokens to Windows Phone devices running the RSA SecurID software token app.

RSA Authentication Manager Administrator's Guide. The *Administrator's Guides* for Authentication Manager 8.x, 7.1, and 6.1 provide an overview of Authentication Manager and its features and describe how to perform administrative tasks, including managing users and security policies and configuring and distributing RSA SecurID tokens.

RSA SecurID Authentication Engine 2.8.1 for Java Developer's Guide. Provides a detailed description of the Java API as well as information for using the API to export token files and authenticate users.

RSA SecurID Authentication Engine 2.3 for C Developer's Guide. Provides a detailed description of the C API as well as information for using the API to export token files and authenticate users.

Support and Service

RSA SecurCare Online	https://knowledge.rsasecurity.com
Customer Support Information	www.emc.com/support/rsa/index.htm
RSA Solution Gallery	https://gallery.emc.com/community/marketplace/rsa?view=overview

RSA SecurCare Online offers a knowledge base that contains answers to common questions and solutions to known problems. It also offers information on new releases, important technical news, and software downloads.

The RSA Solution Gallery provides information about third-party hardware and software products that have been certified to work with RSA products. The gallery includes Secured by RSA Implementation Guides with step-by-step instructions and other information about interoperation of RSA products with these third-party products.

Before You Call Customer Support

Make sure that you have direct access to the computer running the Token Converter.

Please have the following information available when you call:

- ☐ Your RSA Customer/License ID.
- ☐ RSA SecurID Software Token Converter version number.
- ☐ The name of the RSA app that you want to provision with a software token.
- ☐ The make and model of the mobile device on which the problem occurs.
- ☐ The name and version of the mobile operating system under which the problem occurs.

1

Overview

[Introduction](#)

[What's New in Token Converter 3.1](#)

[Supported Mobile Apps](#)

[Supported Operating System Platforms](#)

[Supported Provisioning Servers](#)

Introduction

The RSA SecurID® Software Token Converter 3.1 (Token Converter 3.1) is a command line utility for converting individual RSA SecurID software token files into alternative provisioning formats, including custom compressed token format (CTF) URLs and QR Codes.

Token Converter 3.1 generates custom CTF URLs for all supported RSA SecurID mobile apps. QR Code generation is supported for:

- RSA SecurID Software Token 2.0 for iOS (iOS 2.0 app) and apps built with the RSA SecurID SDK 2.0 for iOS (SDK 2.0 for iOS)

Note: RSA SecurID Software Token 2.0 for iOS supports QR Code provisioning on devices running iOS 7 and later.

- RSA SecurID Software Token 2.0 for Android (Android 2.0 app) and apps built with the RSA SecurID SDK 2.0 for Android (SDK 2.0 for Android)

Token Converter 3.1 does not convert custom CT-KIP URLs into QR Codes. However, you can convert custom CT-KIP URLs for the iOS 2.0 app or the Android 2.0 app using a third-party QR Code conversion tool.

What's New in Token Converter 3.1

The following options are now supported for the iOS 2.0 app and apps built with the SDK 2.0 for iOS:

- `-qr`
- `-prefix`
- `-d`

The following command line options have changed:

- `-ios` has replaced `-iphone`. Use `-ios` when generating custom CTF URLs or QR Codes for the iOS 2.0 and apps built with the SDK 2.0 for iOS.
- `-winphone` has replaced `-mobile`. Use `-winphone` to generate custom CTF URLs for RSA SecurID Software Token 1.0 for Windows Phone (Windows Phone 1.0 app).
- `-f` has been removed. RSA no longer supports the Windows Mobile and Java ME apps that used this option.

Backwards Compatibility

By default, Token Converter 3.1 generates custom CTF URLs using a new format (alphanumeric). To maintain compatibility with older RSA SecurID apps that do not accept the new format, Token Converter 3.1 can generate custom CTF URLs in the legacy format (numeric) used by Token Converter 2.6.1. For more information, see the “`-v`” option on page 16.

Note: Token Converter 2.6.1 functionality is integrated into RSA Authentication Manager 8.x. When provisioning a software token with Authentication Manager 8.x, you can select a CTF distribution option that generates a legacy-format custom CTF URL.

JRE Requirement

Token Converter 3.1 is a Java program. To run the Token Converter, you must have the Java SE Runtime Environment (JRE) installed. The minimum required version is Java SE Runtime Environment 6 Update 45. To download the latest JRE, go to <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

Supported Mobile Apps

RSA SecurID Software Token Converter 3.1 supports the following RSA SecurID mobile apps:

- RSA SecurID Software Token 2.0 for iOS
- RSA SecurID Software Token 1.3 for iPhone and iPad
- Third-party mobile apps that integrate the RSA SecurID SDK for iOS (version 2.0 and 1.5)
- RSA SecurID Software Token for Android (version 2.0 and 1.2)
- Third-party mobile apps that integrate the RSA SecurID SDK for Android (version 2.0 and 1.2)
- RSA SecurID Software Token 1.0 for Windows Phone

Supported Operating System Platforms

RSA SecurID Software Token Converter 3.1 can be run on the following platforms:

- Microsoft Windows Server 2012 (64-bit)
- Microsoft Windows Server 2008 R2 (64-bit)
- Red Hat Enterprise Linux 6 (32-bit/64-bit)
- Red Hat Enterprise Linux 5 (32-bit/64-bit)
- Oracle Solaris 10 (UltraSPARC) (64-bit)

Supported Provisioning Servers

Use one of the following to provision software tokens:

- RSA Authentication Manager 8.x, 7.1, or 6.1
- RSA SecurID Authentication Engine 2.8.1 for Java
- RSA SecurID Authentication Engine 2.3 for C

Important: RSA strongly recommends using token file passwords and binding all software tokens as part of the provisioning process. For more information, see the *Administrator's Guide* for your RSA SecurID software token app.

2

Using RSA SecurID Software Token Converter

[Before You Begin](#)

[Token Converter Input and Output](#)

[Running the Token Converter](#)

[Command Line Usage](#)

[Examples](#)

Before You Begin

Before you run RSA SecurID Software Token Converter 3.1, note the following:

- Token Converter 3.1 is a Java program. To run the Token Converter, you must have the Java SE Runtime Environment (JRE) installed. The minimum required version is Java SE Runtime Environment 6 Update 45. To download the latest JRE, go to <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.
- The Token Converter 3.1 executable is provided as a JAR file. For instructions on using the executable, see “[Running the Token Converter](#)” on page 12.
- SDTID files can contain multiple tokens, but you can only convert one token per SDTID file. If you use a multitoken file, the Token Converter converts the first token and ignores the remaining tokens.
- The Token Converter can convert only one SDTID file at a time.
- RSA strongly recommends password protecting SDTID files used as input to the Token Converter. The converted token data retains the password. The user must enter the password in the RSA SecurID app on the mobile device to import the token.

Important: Password protection must be set up when provisioning SDTID files in RSA Authentication Manager. You cannot add password protection after provisioning SDTID files. If you use Authentication Manager 8.x, you can provision password-protected custom CTF URLs.

- For detailed information on delivering custom CTF URLs to mobile devices running an RSA SecurID app, see the *Administrator's Guide* for the app.
- The iOS 2.0 and Android 2.0 apps have a built-in feature for scanning QR Codes. It is your responsibility to devise your own secure method of delivering QR Codes to these apps or to apps built with the SDK 2.0 for iOS or the SDK 2.0 for Android. RSA places no restrictions on how you deliver QR Codes to your users.

Token Converter Input and Output

RSA SecurID Software Token Converter 3.1 supports the following input and output:

- SDTID file to a custom CTF URL.
- SDTID file to QR Code. (The custom CTF URL is embedded in the QR Code.)

The following table shows the output formats that you can generate for the supported apps.

RSA SecurID App	Custom CTF URL	QR Code
iOS 2.0	✓	✓
iPhone/iPad 1.3	✓	
Third-party apps that integrate SDK 2.0 for iOS	✓	✓
Third-party apps that integrate SDK 1.5 for iOS	✓	
Android 2.0	✓	✓
Android 1.2	✓	
Third-party apps that integrate SDK 2.0 for Android	✓	✓
Windows Phone 1.0	✓	

Running the Token Converter

RSA SecurID Software Token Converter 3.1 is packaged in a zip file for Windows and in a TAR file for UNIX. The package file contains **TokenConverter.jar** and **zxing-core-2.1.jar**.

- **TokenConverter.jar** is responsible for converting token files to custom CTF URLs.
- **zxing-core-2.1.jar** is a third-party SDK used by the Token Converter to convert a custom CTF URL to a QR Code.

Important: When you extract the contents of the package file, you must save both JAR files to the same directory so that you do not need to set a specific CLASSPATH when running the Token Converter.

Run the Token Converter on Windows

1. Unzip the **TokenConverter310.zip** file and extract **TokenConverter.jar** and **zxing-core-2.1.jar** to a directory on your computer. Be sure to extract both files to the same directory.
2. Run **cmd.exe** to open a command line terminal.
3. Change to the directory containing the JAR files.
4. Run **TokenConverter.jar**, using the following format:

```
java -jar TokenConverter.jar filename.sdtid options
```

The token filename must immediately follow `TokenConverter.jar`. Options can be entered in any order. For more information, see [“Command Line Usage”](#) on page 14.

Note: The token files (SDTID files) that you plan to convert do not need to be in the same directory as the JAR files.

Run the Token Converter on UNIX

1. Unzip **TokenConverter310.tar.gz**. The output will be **TokenConverter310.tar**.
2. Untar **TokenConverter310.tar** and extract **TokenConverter.jar** and **zxing-core-2.1.jar** to a directory on your computer. Be sure to save both files to the same directory.
3. Open a command line terminal.
4. Change to the directory containing the JAR files.
5. Run **TokenConverter.jar**, using the following format:

```
java -jar TokenConverter.jar filename.sdtid options
```

The token filename must immediately follow `TokenConverter.jar`. Options can be entered in any order. For more information, see [“Command Line Usage”](#) on page 14.

Note: The token files (SDTID files) that you plan to convert do not need to be in the same directory as the JAR files.

Prerequisite for Using Red Hat Linux

To ensure that the Token Converter will run on Red Hat Linux, you must update the **java.security** file.

1. Open the **\$JAVA_HOME/jre/lib/security/java.security** file in a text editor.
2. Change the line:

```
securerandom.source=file:/dev/./random
```

to

```
securerandom.source=file:/dev/./urandom
```
3. Save the change and exit the text editor.

Command Line Usage

This section describes the Token Converter command line usage.

- Angle brackets identify required options.
- Square brackets identify optional options.

Usage

```
java -jar TokenConverter.jar <token_file> <-android |-ios
|-winphone> [-p <password>] [-d <days_until_expiration>]
[-prefix <custom_output_prefix>] [-v <CTF_version>] [-o
<filename>] [-qr]
```

Usage

```
java -jar TokenConverter.jar -version
```

Platform Target Options

-ios

Generates a custom CTF URL containing CTF data for the iOS platform. The CTF data is generated in the new Token Converter 3.1 format (alphanumeric). RSA delivers CTF data to the iOS app by sending an email containing a custom CTF URL link to the user's device. By default, the Token Converter generates a custom CTF URL for iOS with the following prefix:

```
com.rsa.securid://ctf?ctfData=<CTF_Data>.
```

The scheme portion of the prefix, `com.rsa.securid`, is reserved by RSA for its iOS app.

Use `-v 2` with `-ios` to obtain a custom CTF URL for the iOS 1.3 app. This is required because the iOS 1.3 app only supports legacy-format CTF data.

If you use the SDK 2.0 or 1.5 for iOS to build your own iOS app, and you choose to deliver the CTF data using a custom CTF URL link, use `-prefix` with `-ios` to generate a custom CTF URL prefix. An example of an acceptable scheme is `com.yourco.securid`.

-android

Generates a custom CTF URL containing CTF data for the Android platform. The CTF data is generated in the new Token Converter 3.1 format (alphanumeric). RSA delivers CTF data to the Android app by sending an email containing a custom CTF URL link to the user's device. By default, the Token Converter generates a custom CTF URL for Android with the following prefix:

```
http://127.0.0.1/securid/ctf?ctfData=<CTF_Data>.
```

The first part of the prefix, `http://127.0.0.1/securid`, is reserved by RSA for its Android app.

Use `-v 2` with `-android` to obtain a custom CTF URL for the Android 1.2 app. This is required because the Android 1.2 app only supports legacy-format CTF data.

If you use the SDK 2.0 or 1.2 for Android to build your own Android app, and you choose to deliver CTF data using a custom CTF URL link, use `-prefix` with `-android` to generate a custom CTF URL prefix. An example of an acceptable prefix is `http://yourco.abc.com/securid`.

-winphone

Generates a custom CTF URL for the Windows Phone platform with the reserved RSA prefix `com.rsa.securid://ctf?ctfData=<CTF_Data>`.

Not compatible with `-d` and `-qr`.

Token Options

-p password

Specifies the password used to decrypt an SDTID file and re-encrypt the CTF data. Required for converting password-protected SDTID files. The converted token data retains the password. The user must enter the password in the RSA SecurID app on the device to import the token.

Important: The Token Converter does not support multi-byte characters in the token password, even though RSA Authentication Manager 8.x allows it. If you set a multi-byte token password in Authentication Manager 8.x, you must change it and re-provision the token before using the Token Converter.

-d days-until-expiration

Specifies the number of days from the current date after which a custom CTF URL or QR Code containing CTF data will expire. The supported range is 1 to 99 days. Each day comprises exactly 24 hours. For example, if you use `-d 3` on the command line at 9:30 a.m. on April 12, the custom CTF URL or QR Code expires at 9:30 a.m. on April 15.

`-d` supports the iOS 2.0 app and the Android 2.0 app and mobile apps built with the SDK 2.0 for iOS or the SDK 2.0 for Android. Not compatible with the Windows Phone platform or earlier versions of the iOS or Android apps or SDKs.

If `-d` is not specified, the custom CTF URL or QR Code does not expire until the token expiration date.

CTF Output Formatting Options

-prefix custom_output_prefix

Specifies that the Token Converter will generate a custom CTF URL prefix that uses the text entered as the argument to the `-prefix` option. A sample argument for iOS would be `com.yourco.securid://ctf?ctfData=`.

You must specify a custom prefix for mobile apps that integrate the SDK 2.0 or 1.5 for iOS or the SDK 2.0 or 1.2 for Android.

-v

Specifies the format of the custom CTF URL output.

`-v 2` generates legacy-format CTF data and is required for the iPhone and iPad 1.3 app and the Android 1.2 app, which only accept numeric CTF data.

Not compatible with `-d` and `-qr`.

File Output Options

-o filename

Writes the Token Converter output to a file. If `-o` is not specified, output is written to the screen. To write an output file to a location other than the current directory, you must specify either a relative or absolute file path after `-o`.

When generating a custom CTF URL, give the output filename a `.txt` extension.

The default extension for QR Code output is `.jpeg`. If you use `.jpg`, the output file retains that extension. The Token Converter accepts either extension.

-qr

Generates a custom CTF URL embedded in a QR Code. Supported with the iOS 2.0 app and the Android 2.0 app.

Note: The iOS 2.0 app supports provisioning QR Codes to devices running iOS 7 or later.

`-qr` is also supported with the SDK 2.0 for iOS and the SDK 2.0 for Android. These SDKs do not contain methods for importing QR Codes. To support QR Code provisioning, apps created with these SDKs must build in functionality to scan QR Codes.

Use `-ios` or `-android` with `-qr`, as appropriate.

`-o` is required with `-qr`.

Compatible with `-d`.

- version

Displays the Token Converter version on the screen. The value for Token Converter 3.1.0 is "3.1.0."

Examples

The following sections provide sample commands.

Note: The examples are for Windows operating systems. If you are running the Token Converter on a UNIX system, adjust the command line accordingly.

Token Converter Version

This example displays the Token Converter version to the screen.

Input

```
java -jar TokenConverter.jar -version
```

Output

```
Token Converter 3.1.0
```

iOS 2.0 App

SDTID file to custom CTF URL

This example generates a custom CTF URL that starts with the reserved RSA scheme. The input file is password protected. The custom CTF URL is written to the output file specified by -o. The custom CTF URL expires in three days.

Input

```
java -jar TokenConverter.jar c:\tmp\user2-passwordtoken.sdtid  
-ios -p tokenpw1 -o c:\tmp\tokenfile.txt -d 3
```

Output

```
com.rsa.securid://ctf?ctfData=AwAAMYK8JX0Y5c1BPzYQCboKes7JLuID2  
XCY1D08K3XJ62cxgrwlfRjlyUE%2FNhAJs4p6zsku4gPZcJiUM7wrdcnrZ%2Fur  
qwHoT3D3gDhWm2mGT1Wxc7L3qz8w6%2Bk4m5DbKK4muOM%2BbxVQoLo71U2baUQ  
jul%2Fj%2B1YYtyYYGQLdJxjYo2KbRayCFNdeWE5KGOnpKu95I1fmm7eNvBiNQ7  
ueauj%3QEoI4zPxxn7iIDCE6Mvwh9LAT2s84maGqYIfUVVCf65ErB9BuSRkK6%2  
FMqfDc7WdTBGPbeo4gpKRvw07yQRxKNENO7Etoi1jAklfJy1G1pr3LuiivTz2b1M  
lIw%2BhuQwsKUSBAslFgrl7dYGMmz6PN1gkY1586WMPJkSGABupTFul2%2F%2B
```

SDTID file to QR Code

This command generates a custom CTF URL embedded in a QR Code. The input file is password protected. The QR Code expires in 3 days. The QR Code is written to the output file specified by -o.

Input

```
java -jar TokenConverter.jar c:\tmp\token1.sdtid -ios -qr  
-p tokenpw1 -o c:\tmp\qrtoken.jpeg -d 3
```

iOS 2.0 SDK

SDTID file to custom CTF URL with custom prefix

This example generates a custom CTF URL that starts with a custom scheme (in this example, `com.yourco.securid`). The custom prefix is specified as the argument to the `-prefix` option. The input file is password protected. The custom CTF URL expires in 7 days. The custom CTF URL is written to the output file specified by `-o`.

Input

```
java -jar TokenConverter.jar c:\tmp\token1.sdtid -ios
-p t0kenpw1 -prefix com.yourco.securid://ctf?ctfData= -o
c:\tmp\tokenfile.txt -d 7
```

Output

```
com.yourco.securid://ctf?ctfData=AwAAGKCsXaTYtDJ4iQGFH84mt0nCUC
X4fNSuE7bh7mPzjsYYoKxdpNi0MniJAYUfzia3ScJQJfh81K4TtuHuY%2FOOxk8
b9EaDHYuMxtHqOnB39TU1JA8OkTOhh%2B%2BCqKl16qK%2FS%2BzQ1FVuilNzLC
Q9RaJz4Uv1N1CUKae0UNQJakYqH4UQLUnspbzsFQEH6JXSFUroB4DnuYPidwerG
22fOQj6%2FxWiVrz3kh0qs1Mhapkp5o0pb5Wsd0kWYi%2F3S9AlHyc9u0I%2FCW
3NdZPsz2Py832dJOkn9fccvE8%2B2HDBfyh6uDStuIEKXJgNmh5Ji3r3wuVMgGT
vGteNNcBobFRz%2FFFtnKNtc0QLVogEuepDSO5T8uP8XpCvOFgWbUsfsRZ%2BEN
Tt
```

SDTID file to QR Code with custom prefix

This command generates a custom CTF URL embedded in a QR Code. The custom prefix, which starts with a custom scheme (in this example, `com.yourco.securid`), is specified as the argument to the `-prefix` option. The input file is password protected. The QR Code is written to the output file specified by `-o`.

Input

```
java -jar TokenConverter.jar c:\tmp\token1.sdtid -ios -qr
-p t0kenpw1 -prefix com.yourco.securid://ctf?ctfData= -o
c:\tmp\qrtoken.jpeg
```

iOS (iPhone and iPad) 1.3 App

SDTID file to Custom CTF URL

This example generates a custom CTF URL that starts with the reserved RSA scheme. `-v 2` is required, because the iOS 1.3 app only supports legacy-format CTF strings. The input file is password protected. The custom CTF URL is written to the output file specified by `-o`.

Input

```
java -jar TokenConverter.jar c:\tmp\user2-passwordtoken.sdtid
-ios -p t0kenpw1 -o c:\tmp\tokenfile.txt -v 2
```

Output

```
com.rsa.securid://ctf?ctfData=200010000047343660247556373456223
35363640565103593067284028475474176266671632140
```

iOS (iPhone and iPad) 1.5 SDK

SDTID file to Custom CTF URL with custom prefix

This example generates a custom CTF URL that starts with a custom scheme (in this example, `com.yourco.securid`). The custom prefix is specified as the argument to the `-prefix` option. `-v 2` is required, because the iOS 1.5 SDK only supports legacy-format CTF strings. The input file is password protected. The custom CTF URL is written to the output file specified by `-o`.

Input

```
java -jar TokenConverter.jar c:\tmp\token1.sdtid -ios -p
t0kenpw1 -prefix com.yourco.securid://ctf?ctfData= -o
c:\tmp\tokenfile.txt -v 2
```

Output

```
com.yourco.securid://ctf?ctfData=200010000047343660247556373465
432178905436751034050145664172022474176266671632140
```

Android 2.0 App

SDTID file to custom CTF URL

This example generates a custom CTF URL that starts with the reserved RSA prefix. The input file is password protected. The CTF string is written to the output file specified by -o. The CTF string expires in three days.

Input

```
java -jar TokenConverter.jar c:\tmp\user2-passwordtoken.sdtid
-android -p t0kenpw1 -o c:\tmp\tokenfile.txt -d 3
```

Output

```
http://127.0.0.1/securig/ctf?ctfData=AwAAMYK8JX0Y5clBPzYQCbOKes
7JLuID2XCY1DO8K3XJ62cxgrwlfRjlyUE%2FNhAJS4p6zsku4gPZcJiUM7wrdsn
rZ%2FurqwHoT3D3gDhWm2mGT1Wxc7L3qz8w6%2Bk4m5DbKK4muOM%2BbxVQoLo7
1U2baUQjul%2Fj%2B1YYtyYYGQLdJxjYo2KbRayCFNdeWE5KGOnpKu95I1fmm7e
NvBiNQ7ueauj%3QEoI4zPxxn7iIDCE6Mvwh9LAT2s84maGqYIfUVVCf65ErB9Bu
SRkK6%2FMqfDc7WdTBGPbeo4gpKRvw07yQRxKNENO7EtoiljAklfJy1G1pr3Lui
vTz2b1MlIw%2BhuQwsKUSBAslFgrl7dYGMmz6PN1gkY1586WMPJkSGABupTFul2
%2F%2B
```

SDTID file to QR Code

This command generates a custom CTF URL embedded in a QR Code. The input file is password protected. The QR Code expires in 3 days. The QR Code is written to the output file specified by -o.

```
java -jar TokenConverter.jar c:\tmp\token1.sdtid -android -qr
-p t0kenpw1 -o c:\tmp\qrtoken.jpeg -d 3
```

Android 2.0 SDK

SDTID file to custom CTF URL with custom prefix

This example generates a custom CTF URL that starts with a custom prefix (in this example, `http://yourco.abc.com/securig`). The custom prefix is specified as the argument to the `-prefix` option. The input file is password protected. The CTF string expires in 7 days. The CTF string is written to the output file specified by `-o`.

Input

```
java -jar TokenConverter.jar c:\tmp\token1.sdtid -android
-p t0kenpw1 -prefix http://yourco.abc.com/securig/ctf?ctfData=
-o c:\tmp\tokenfile.txt -d 7
```

Output

```
http://yourco.abc.com/securig/ctf?ctfData=AwAAGKCsXaTYtDJ4iQGfH
84mt0nCUCX4fNSuE7bh7mPzjsYYoKxdpNi0MniJAYUfzia3ScJQJfh81K4TtuHu
Y%2FOOxk8b9EaDHuMxtHqOnB39TU1JA8OkTOhh%2B%2BCqKl16qK%2FS%2BzQ1
FVuilNzLCQ9RaJz4Uv1N1CUKae0UNQJakYqH4UQLUnspbzsfQEH6JXSFUroB4Dn
uYPidwerG22fOQj6%2FxiVrz3kh0qs1Mhapkp5o0pb5Wsd0kWi%2F3S9AlHyc
9u0I%2FCW3NdZPs2Py832dJOkn9fccvE8%2B2HDBfyh6uDStuIEKXJgNmh5Ji3
r3wuVMgGTvGteNNcBobFRz%2FFFtnKNtc0QLVogEuepDS05T8uP8XpCvOFgWbUs
fsRZ%2BENTt
```

SDTID file to QR Code with custom prefix

This command generates a custom CTF URL embedded in a QR Code. The custom prefix (which, in this example, starts with `http://yourco.abc.com/securig`), is specified as the argument to the `-prefix` option. The input file is password protected. The QR Code is written to the output file specified by `-o`.

```
java -jar TokenConverter.jar c:\tmp\token1.sdtid -android -qr
-p t0kenpw1 -prefix http://yourco.abc.com/securig/ctf?ctfData=
-o c:\tmp\qrtoken.jpeg
```

Android 1.2 App

SDTID file to Custom CTF URL

This example generates a custom CTF URL that starts with the reserved RSA prefix. `-v 2` is required, because the Android 1.2 app only supports legacy-format CTF data. The input file is password protected. The CTF string is written to the output file specified by `-o`.

Input

```
java -jar TokenConverter.jar c:\tmp\user2-passwordtoken.sdtid
-android -p t0kenpw1 -o c:\tmp\tokenfile.txt -v 2
```

Output

```
http://127.0.0.1/securid/ctf?ctfData=200010000004734366024755637
3456223353636405651034050145664172022474176266671632140
```

Android 1.2 SDK

SDTID file to Custom CTF URL with custom prefix

This example generates a custom CTF URL that starts with a custom prefix (in this example, `http://yourco.abc.com/securid`). The custom prefix is specified as the argument to the `-prefix` option. `-v 2` is required, because the Android 1.2 SDK only supports legacy-format CTF strings. The input file is password protected. The CTF string is written to the output file specified by `-o`.

Input

```
java -jar TokenConverter.jar c:\tmp\token1.sdtid -android
-p t0kenpw1 -prefix http://yourco.abc.com/securid/ctf?ctfData=
-o c:\tmp\tokenfile.txt -v 2
```

Output

```
http://yourco.abc.com/securid/ctf?ctfData=2000100000047343660247
556373456223353636405651034050145664172022474176266671632140
```

Windows Phone 1.0 App

SDTID file to Custom CTF URL

This example generates a custom CTF URL that starts with the reserved RSA scheme for Windows Phone. `-v 2` is required, because the Windows Phone app only supports legacy-format CTF strings. The input file is password protected. The custom CTF URL is written to the output file specified by `-o`.

Input

```
java -jar TokenConverter.jar c:\tmp\user2-passwordtoken.sdtid  
-winphone -p tokenpw1 -o c:\tmp\tokenfile.txt
```

Output

```
com.rsa.securid://ctf?ctfData=200010000047343660247556373456223  
35363640565103593067284028475474176266671632140
```


3

Error Messages

Error Messages

Error Messages

The following table lists the errors codes and messages returned by the Token Converter.

Error Code	Message	Comments
0	N/A	The conversion was successful.
1	Error: Invalid output file parameter specified after -o.	No output file, or an invalid output file, was specified after -o.
2	Error: No token file password specified after -p.	Must specify token file password after -p.
3	Error: Invalid CTF version number specified after -v. Must specify a value of 2.	-v 2 must be specified for the Android 1.2 or iPhone and iPad 1.3 platform.
4	Error: No parameter specified after java -jar TokenConverter.jar	Either an SDTID file name or -version must be specified.
5	Error: Invalid parameter specified.	General command line syntax error. For example, an invalid argument was specified after -android.
6	Error: Invalid token type.	The token type is not supported (for example, a 64-bit token).
7	Error: Password required.	The token file is password protected, but no password or an incorrect password was entered on the command line.
8	Error: Invalid token record.	An invalid SDTID file was specified.
9	Error: Failed to generate CTF string or QR Code.	An internal error occurred.
10	Error: Cannot write to output file.	The Token Converter cannot access the specified output file. For example, the directory is write protected or the path is incorrect.

11	Error: Unsupported option.	-f, -mobile, or -iphone was specified. These options are not supported in Token Converter 3.1.
12	Error: -winphone does not support -qr. Error: -winphone does not support -d.	An unsupported option was specified with -winphone.
13	Error: Invalid parameter specified after -prefix.	No prefix string, or an invalid prefix string, was specified after -prefix.
14	Error: Invalid parameter specified after -d.	No value was specified after -d, or the value was not between 1-99.
15	Error: -o filename not specified with -qr.	-o and a file name are required with -qr.
16	Error: Invalid input file.	The Token Converter cannot access the specified input file. There is no read access or the file does not exist.
17	Error: Only one target platform can be specified. Choose either -android, -ios, or -winphone. Error: At least one target platform must be specified. Choose either -android, -ios, or -winphone.	These errors are generated if the user does not specify -android, -ios, or -winphone, or if more than one target is specified.
