



SDG CONVERSION TOKEN SERVICE

2.3.x INSTALLATION/CONFIGURATION – STANDALONE

sdgc.com

© 2026 SDG Corporation
All Rights Reserved

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by SDG Corporation at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of SDG. This Documentation is confidential and proprietary information of SDG and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and SDG governing your use of the SDG software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and SDG.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all SDG copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to SDG that all copies and partial copies of the Documentation have been returned to SDG or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, SDG CORPORATION PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL SDG BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF SDG IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is SDG Corporation.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) – (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2026 SDG Corporation. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contact Information:

SDG Corporation

U.S. Global Headquarters
75 North Water St. Norwalk, Connecticut, 06854
United States
Phone: +1 (203) 866-8886

SDG Global Tech Center

India (Country HQ)
A-10, Sector 2
Noida, UP
India 201301
Phone: +91 120-4014000

SDG Canada

9131 Keele Street, Suite A4
Vaughan, ON L4K 0G7
Canada

Third Party Libraries and Licensing

Library	Version
Jersey	2.35
jackson	2.11.4
googlehttp	1.38.0
jetty	9.4.44.v20210927
googleOauth	1.33.0
jsonwebtoken	0.9.1
commons-codec	1.15
slf4j	2.0.0-alpha7
springframework	5.3.20
jasyp	1.9.3
commons-io	2.11.0

Jersey – RESTFul Web Services in Java

Used under the GPL License

<https://eclipse-ee4j.github.io/jersey.github.io/license.html>

Jackson

Used under an Open Source license from Apache

<https://www.apache.org/licenses/LICENSE-2.0>

GoogleHTTP

Used under Open Source GPL Licensing

<https://opensource.google/documentation/reference/thirdparty/licenses>

Jetty

Used under the Eclipse Public License and Apache 2.0 License

<http://www.eclipse.org/jetty/licenses.php>

Google OAuth

Used under Open Source GPL Licensing

<https://opensource.google/documentation/reference/thirdparty/licenses>

JSON Web Token

Licensed under the Apache 2.0 License

<http://www.eclipse.org/jetty/licenses.php>

Commons Codec

Used under an Open Source license from Apache
<https://www.apache.org/licenses/LICENSE-2.0>

SLF4J

Used under open source GPL Licensing
<https://www.slf4j.org/license.html>

Spring Framework

Used under the Apache 2.0 Open Source License
<https://www.apache.org/licenses/LICENSE-2.0>

JASPYT

Used under the Apache 2.0 Open Source License
<http://www.jaspyt.org/license.html>

Commons-IO

Used under the Apache 2.0 Open Source License
<https://www.apache.org/licenses/LICENSE-2.0>

Table of Contents

Chapter 1: Overview.....	7
SDG Conversion Token Service (CTS) Overview.....	7
Workflow Integration with Ping Federate.....	7
Platform Support.....	8
Components of the SDG Conversion Token Service.....	8
Component Location.....	9
Which Install Path Should You Choose?.....	9
Chapter 2: SDG Conversion Token Service Standalone Install for SiteMinder.....	11
Install Package Overview.....	11
CTS Directory Structure.....	11
SDG Conversion Token Service File Details.....	12
Creation of the CA SiteMinder Policy Server Objects.....	16
SiteMinder Policy Base Object Creation.....	16
Update the CTS Domain for Your Environment.....	18
Configure Additional Agent Config Object Settings.....	20
CA SiteMinder Java SDK and Policy Server Process.....	20
Register the SDK Agent Used by the SDG Conversion Token Service.....	21
Certificate & KeyStore for Mutual Authentication.....	22
SDG Conversion Token Service Configuration Parameters.....	22
cts.properties File.....	22
jetty.xml File.....	23
Chapter 3: Starting the CTS Service.....	24
Chapter 4: Configuring Mutual Client Certificate Authentication.....	25
Client Side Certificate Generation.....	25
Generate a Client Certificate Key Pair.....	25
Export client certificate without Private Key for use on server side (.cer file).....	26
Export the Certificate for Use on the CTS Server.....	26
Convert the Certificate to pem Format.....	27
Extract the Private Key.....	27
Files Generated.....	28
Server Side Certificate Generation and Mutual SSL Configuration.....	28
Generate a Keystore Key Pair.....	28
Copy the Client Certificate to the Server.....	29
Import the Client Certificate into the keystore as a Trusted Certificate	29
Verify the Certificate in the Keystore.....	30
Add Client Certificate Serial Number to the authenticateusers.txt File.....	31

CHAPTER 1: OVERVIEW

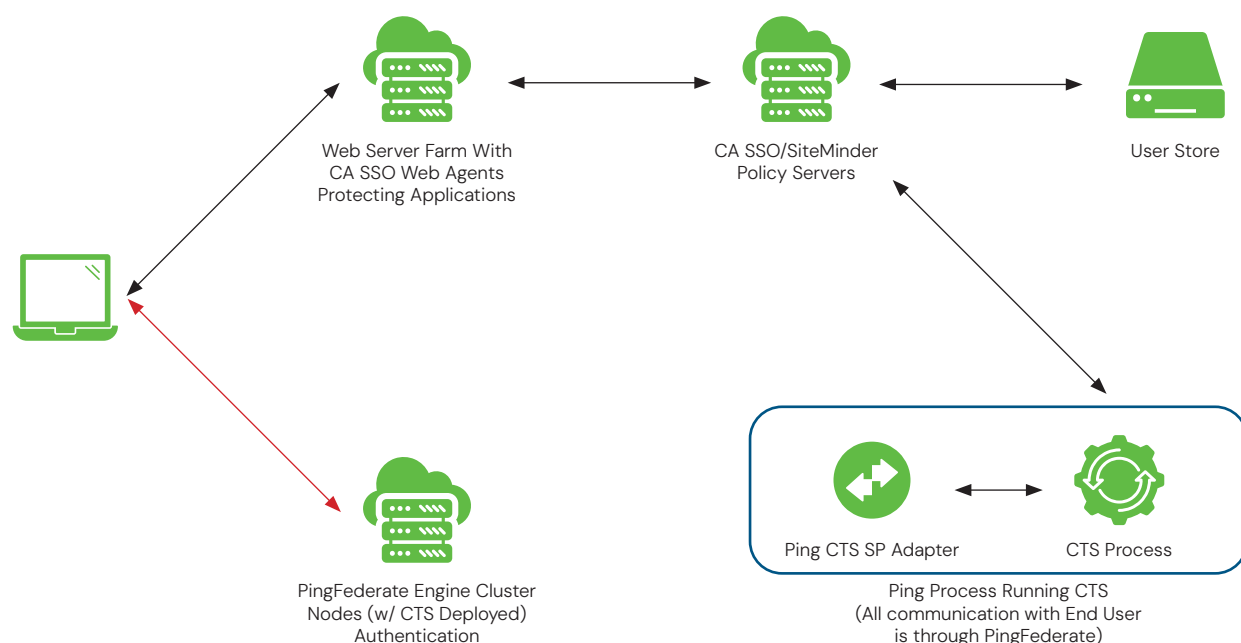
SDG Conversion Token Service (CTS) Overview

The SDG Conversion Token Service (CTS) is a Java REST Web Service that allows the exchange of tokens between different security platforms. In this installation CTS is used to exchange tokens between CA SiteMinder/SSO and PingFederate.

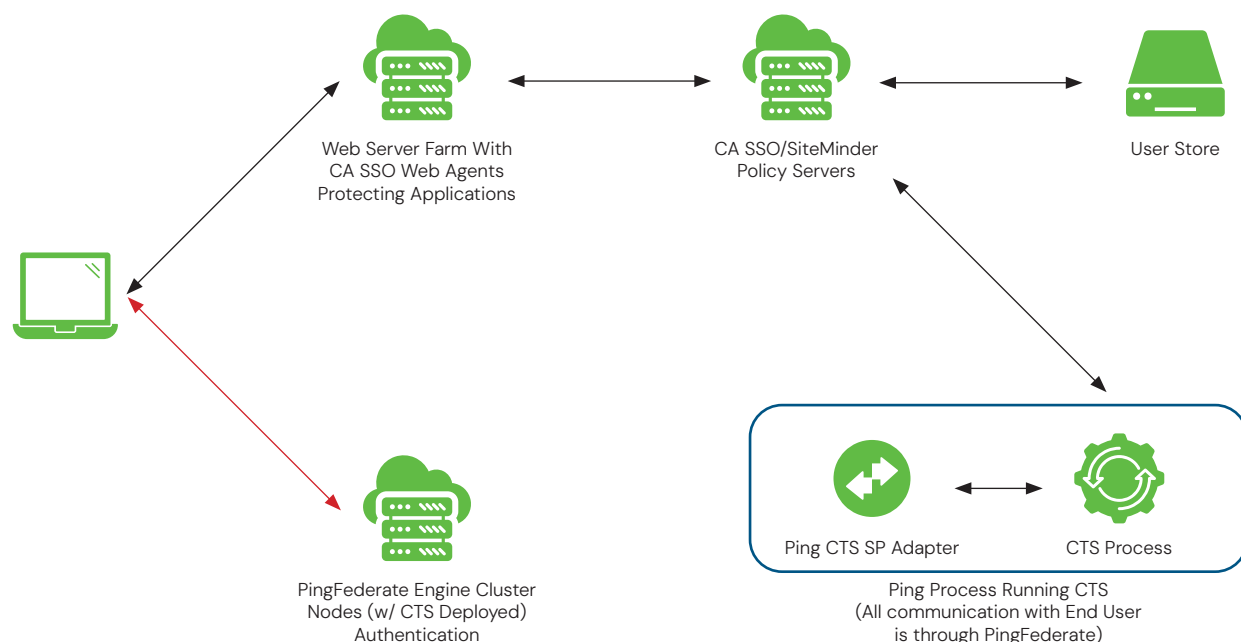
It is built with the CA SSO SDK that allows it to interact with CA SiteMinder Policy Servers. It supports a level of communication that a full featured web agent leverages to talk to multiple policy servers and supports round robin and failover communication.

Workflow Integration with Ping Federate

The SDG Conversion Token Service can be deployed in two different architectures to interoperate with PingFederate. Whether you deploy CTS to run within the PingFederate Jetty Instance or in its Standalone configuration CTS works with PingFederate's SDG Conversion Token Adapters to implement the following use cases.



Use Case 1 – User authenticates to a CA SSO/SiteMinder Resource, existing or newly created SMSESSION Tokens can then be exchanged for access to a Ping Federate protected application with no additional challenge through the use of the SDG Conversion Token Service allowing single sign on to both platforms.



Use Case 2 – User starts at PingFederate and accesses an Application Definition with an Auth Tree with the SDG SP Adapter defined that will send the user to a CA SSO/SiteMinder protected application. The Ping CTS SP Adapter provides trusted information about the user and generates an SMSESSION based on this information. The user now has valid sessions and work between both platforms.

Platform Support

PingFederate Versions	SiteMinder Versions (includes all Service Packs)	Java Versions Validated
<ul style="list-style-type: none"> PingFederate 9.x PingFederate 10.x 	<ul style="list-style-type: none"> SiteMinder 12.52 SiteMinder 12.6 SiteMinder 12.7 SiteMinder 12.8 	<ul style="list-style-type: none"> Oracle Java 1.8 Oracle Java 11* Amazon Corretto 11* OpenJDK 11*

* The SiteMinder Agent SDK is only supported on Oracle 1.8, Oracle (formerly Sun) 1.7, and IBM Java 1.7.x & 1.8.x. SDG cannot guarantee support of other versions of Java since they are unsupported by Broadcom.

Components of the SDG Conversion Token Service

The following components are required as part of the collective solution:

- SDG Conversion Token Service (SDG)
- CTS Integration Kit (Ping Identity)
- Ping Federate (Ping Identity)
- CA SiteMinder (CA/Broadcom)
- CA SiteMinder SDK JARS (Included with CTS)

SDG Conversion Token Service
CTS Integration Kit (Ping Identity)
CA SiteMinder SDK (Included with our kit)
CA SiteMinder (CA/Broadcom)



PingFederate Engine Node

SDG Conversion Token Service
CTS Integration Kit (IDP/SP Adapter JARS)
CA SiteMinder SDK (JARS)



CA SiteMinder Policy Server

CTS Domain/Policy Objects
CTS Agent Configuration Objects
CTS Host Configuration Object
CTS Trusted Host



PingFederate Admin Node

CTS Integration Kit (IDP/SP Adapter Jars)
CA SiteMinder SDK Library Files (Jars)

Component Location

Since the SDG Conversion Token Service requires the use of multiple components please refer to the following table and diagram to plan your installation accordingly.

PingFederate	SDG Conversion Token Service WAR	CA SSO SDK JAR Files	CTS Adapter
Required on PF Stand-Alone	Yes	Yes	(Yes) Configure IDP or SP Adapter as needed
Required on PF Engine	Yes	Yes	(Yes) Configure IDP or SP Adapter as needed
Required on PF Admin Only	No	No	(Yes) Configure IDP or SP Adapter as needed



**PingFederate Engine
Node(s) or StandAlone**

- SDG Conversion Token Service WAR
- cts.properties
- authenticatedUsers.txt
- cbSmHost.conf
- smagentapi.jar
- crypto.jar
- Edit run.properties
- Edit jettyruntime.xml
- SDG Integration Kit JAR



**PingFederate
Administrator Console**

- SDG Integration Kit JAR

Which Install Path Should You Choose?

The SDG Conversion Token Service can be deployed in one of two scenarios, depending on your requirements and infrastructure layout of CA SiteMinder and PingFederate.

If you have a single CA SiteMinder environment to support (B2E) then you should go to Chapter 2 and do the Embedded Installation of CTS in PingFederate. There is a limit that you can only run **ONE** instance of CTS within your PingFederate Installation. All the steps to setup CTS with PingFederate are in that Chapter.

Now, if your infrastructure needs and requirements are that you need to talk to multiple CA SSO instance (B2E, B2B, B2C) from a single PingFederate Infrastructure you will need to **MULTIPLE** Standalone instances of CTS. This is best explained in Chapters 3, 4 and 5. You can run multiple instances of CTS on the same server, you just need to edit the jetty-runtime.xml and change each instance to it's own port.

CHAPTER 2: SDG CTS STANDALONE INSTALL FOR SITEMINDER

This section contains the following topics:

- Install package overview
- CTS directory structure
- Creation of CA SiteMinder Policy Server objects
- CA SiteMinder SDK installation and setup
- Certificate and Keystore for mutual authentication
- CTS configuration parameters

Install Package Overview

Unzip the contents of the cts.zip file into a designated location from where the SDG Conversion Token Service will be executed.

The install is packed in the following structure:

Folder Name	Description of Contents
cts	Contains the main war file, start script and config and lib directories
Sample Client Certificate	Contains sample client certificates
script	Contains a perl script that is used to create the SiteMinder base objects on the Policy Server

CTS Directory Structure

Once unzipped, the CTS structure is outlined in the following table:

Name	Type	Description
config	Folder	Contains the configuration files that is used by CTS
lib	Folder	Contains the SiteMinder SDK library files
run.sh / stop.sh / run.bat	File	Shell script to run or stop CTS
sdg-conversiontokenservice-combined<version number>.war	File	CTS war file
config/authenticatedUsers.txt	File	Contains a list of certificate serial numbers for mutual authentication
config/SmHost.conf	File	The smhost.conf file generated by running the smreg-host.sh / smreghost.bat script. Note: The name of this file needs to be referenced in the cts.properties file
config/cts.properties	File	The main configuration file that is used by CTS when it is deployed as a standalone application
config/cts_server.keystore	File	Contains the certificate keystore. Note: The name of this file needs to be referenced in the jetty.xml file

Name	Type	Description
config/jetty.xml	File	Standard Jetty.xml file Defines the server configuration when running as a standalone CTS application. Parameters like Port, keystore, password to the keystore, ssl are defined in this file.
config/Allowed_Addresses.xml	File	XML file containing a whitelist of IP addresses that can connect to CTS. An empty list disables this feature.
config/OAuth2_providers.xml	File	XML file defining a list of OAuth 2.0 providers that can be used for 3rd party OAuth based authentication.
config/Protected_Resources.xml	File	XML File defining a list of SSO protected resources that can be accessed after successful authentication with an OAuth 2.0 provider and SSO.

SDG Conversion Token Service File Details

Once unzipped, the CTS structure is outlined in the following table:

```
JAVA_HOME=/apps/Java/jdk
CTS_HOME=/apps/CA/sdg-conversiontokenservice-1.0/cts_2_0
LOG_FILE=$CTS_HOME/logs/cts_start.log nohup $JAVA_HOME/bin/java -DINFO
-cp
$CTS_HOME/config:$CTS_HOME/lib/smagentapi.jar:$CTS_HOME/lib/cryptoj.jar:$CTS_HOME/sdgconversiontokenservice-
combined-2.0.war:. Launch $CTS_HOME/config/jetty.xml >$CTS_HOME/cts_start.log 2>&1 &
```

Contents of run.bat:

```
java -cp config\lib\smagentapi.jar;lib\cryptoj.jar;sdg-conversiontokenservice-combined-2.3.0.war;. Dhttps.protocols=TLSv1.2
Launch config\jetty.xml
```

Example contents of authenticatedUsers.txt:

```
00C33C6C513105A65A
013BOA151EDA
```

Example contents of cbSmHost.conf:

```
# Host Registration File – cbSmHost.conf
#
# This file contains bootstrap information required by
# the SiteMinder Agent API to connect to Policy Servers
# at startup. Be sure the IP addresses and ports below #
identify valid listening Policy Servers. Please do not #
hand edit the encrypted SharedSecret entry.
hostname="tokenservices_hn" hostconfigobject="tp_hco"
policyserver="172.16.246.181,44441,44442,44443" requesttimeout="30"
sharedsecret="{RC2}2gll/+2JdSiPhGHFxbnVrGAla2LaetSO8ChdOyUxoy19XKfaMABkokyPtP8cDk4F fwNORZ4ycsFg6K-
mYUpR+mfgNXsGikcwIMhJNvph2TmA/F4Ql2G7PqyZ8ft79LmCYJqevtUt34lZxQp k fOXR+4uRJjggEEHGOZzRNsoKx-
z+O8Vr22LT4500AoCEIOGO" sharedsecrettime="1353116491" fipsmode="COMPAT"
```

Example Contents of jetty.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//DTD Configure//EN" "http://www.eclipse.org/jetty/configure_9_3.
dtd">
<Configure id="server" class="org.eclipse.jetty.server.Server">
  <!-- ===== -->
  <!-- Http Configuration. -->
  <!-- This is a common configuration instance used by all -->
  <!-- connectors that can carry HTTP semantics (HTTP, HTTPS, etc.)-->
  <!-- It configures the non wire protocol aspects of the HTTP -->
  <!-- semantic. -->
  <!-- -->
  <!-- This configuration is only defined here and is used by -->
  <!-- reference from other XML files such as jetty-http.xml, -->
  <!-- jetty-https.xml and other configuration files which -->
  <!-- instantiate the connectors. -->
  <!-- -->
  <!-- Consult the javadoc of o.e.j.server.HttpConfiguration -->
  <!-- for all configuration that may be set here. -->
  <!-- ===== -->
  <New id="httpConfig" class="org.eclipse.jetty.server.HttpConfiguration">
    <Set name="secureScheme">https</Set>
```

```

    <Set name="securePort"><Property name="jetty.ssl.port" default="8443" />9596</Set>
    <Set name="outputBufferSize">32768</Set>
</New>

<!-- ===== -->
<!-- SSL Context Factory for HTTPS -->
<!-- SSL requires a certificate so we configure a factory for ssl contents -->
<!-- with information pointing to what keystore the ssl connection needs -->
<!-- to know about. Much more configuration is available the ssl context, -->
<!-- including things like choosing the particular certificate out of a -->
<!-- keystore to be used. -->
<!-- ===== -->
<New id="sslContextFactory" class="org.eclipse.jetty.util.ssl.SslContextFactory">
    <Set name="keyStorePath">config/cts_server.keystore</Set>
    <Set name="keyStorePassword">keystorepassword</Set>
    <Set name="keyManagerPassword">keystorepassword</Set>
    <Set name="trustStorePath">config/cts_server.keystore</Set>
    <Set name="trustStorePassword">keystorepassword</Set>
    <Set name="protocol">TLSv1.2</Set>
    <Set name="wantClientAuth">true</Set>
</New>

<New id="src" class="org.eclipse.jetty.server.SecureRequestCustomizer">
<Set name="stsMaxAge">2000</Set>
    <Set name="stsIncludeSubDomains">true</Set>    </New>

<!-- ===== --> <!-- HTTPS Configuration-->
<!-- A new HttpConfiguration object is needed for the next connector and -->
<!-- you can pass the old one as an argument to effectively clone the -->
<!-- contents. On this HttpConfiguration object we add a -->
<!-- SecureRequestCustomizer which is how a new connector is able to -->
<!-- resolve the https connection before handing control over to the Jetty -->
<!-- Server. -->
<!-- ===== -->

<New id="httpsConfig" class="org.eclipse.jetty.server.HttpConfiguration">
<Arg>
    <Ref refid="httpConfig"></Ref>
</Arg>

```

```
</New>

<New id="https" class="org.eclipse.jetty.server.ServerConnector">
  <Arg name="server"><Ref refid="server"></Ref></Arg>
  <Arg name="factories">
    <Array type="org.eclipse.jetty.server.ConnectionFactory">
      <Item>
        <New class="org.eclipse.jetty.server.SslConnectionFactory">
          <Arg name="sslContextFactory"><Ref refid="sslContextFactory"></Ref></Arg>
        <Arg name="next">HTTP/1.1</Arg>
      </New>
    </Item>
  </Array>
  <Item>
    <New class="org.eclipse.jetty.server.HttpConnectionFactory">
      <Arg>
        <Ref refid="httpsConfig"></Ref>
      </Arg>
    </New>
  </Item>
</Item>
</Array>
</Arg>
<Set name="port">9596</Set>
<Set name="IdleTimeout">30000</Set>
</New>

<!-- ===== -->
<!-- extra server options -->
<!-- ===== -->
<Set name="connectors">
  <Array type="org.eclipse.jetty.server.Connector">
    <Item>
      <Ref refid="https"></Ref>
    </Item>
  </Array>
</Set>
</Configure>
```

Creation of the CA SiteMinder Policy Server Objects

The SDG Conversion Token Service CA SiteMinder connector requires a set of policies to exist in the environment. In addition to the configuration steps outlined here, a User Directory and a Host Configuration Object must exist. The items are not created in this section.

There are two steps for configuring the SiteMinder policies:

1. Run the cts-install.pl script
2. Update the CTS Domain for your environment
3. Configure additional Agent Config Object settings

SiteMinder Policy Base Object Creation

The cts-install.pl Perl script is used to create the base SiteMinder policies leveraged by the CTS SiteMinder connector. The following steps outline the process for creating the base objects:

1. Copy the <CTS Home>\script\cts-install.pl or <CTS Home>/scripts/cts-install.pl file to the Policy Server
2. Run the Perl Script to create the SiteMinder Policy Server Objects using the following command
(NOTE: SiteMinder's CLI Perl installation should be used to run this command in <SiteMinder Policy Server Home>\CLI\bin or <SiteMinder Policy Server Home>/CLI/bin directory): perl cts-install.pl
3. The installer begins
4. When prompted, enter the username and password of a siteminder admin user (e.g. siteminder):

```
***** SDG Conversion Token Service Policy Server Installer *****
```

```
Enter Policy Server Administrator credentials
```

```
----- Administrator
```

```
ID: siteminder
```

```
Administrator Password: <siteminder password>
```

5. The CTS uses a SiteMinder Domain to create its objects. The Domain name must be unique. Enter the name of the Domain to be created:

```
Enter unique CTS Domain Name
```

```
-----
```

```
Name (or X to exit): SDG Conversion Token Service Domain
```

6. An existing user directory is configured for authentication of user tokens. Specify the user directory to authenticate users that will be using CTS:

Select User Directory Used for Authenticating Users

----- [1] FederationWSCustomUserStore

[2] SAML2FederationCustomUserStore

[3] FedBCCustomUserStore

[4] FedBCCertUserDirectory

[5] **Demo User Directory**

[X] Exit install script

Enter number or X to exit: 5

7. For username tokens, a search lookup is required to locate the user identity in the user directory. For details on the format of this lookup, refer to the CA SiteMinder Windows Authentication Scheme documentation. Specify the user search lookup:

Enter the user lookup search query for locating users

Use %{UID} for the user ID

Optionally use %{DOMAIN} to further restrict the search

For example: (sAMAccountName=%{UID})

----- User lookup (or X to exit): (uid=%{UID})

8. Confirm the installation parameters:

Confirm Installation Parameters

[1] CTS Domain Name: SDG Conversion Token Service Domain

[2] Selected User Directory: Demo User Directory

[3] User search query: (uid=%{UID})

Enter [Y]es to continue, [X] to exit or number to modify value: y

9. The installation begins:

Creating CTS objects...Creating CTS Agent Identity...Done

Creating CTS User Attribute Authentication Scheme...Done

Creating CTS Agent Configuration Object...Done

Adding CTS Agent Configuration Parameters...Done

Associating User Directory object...Done
Creating CTS Configuration Domain – SDG Conversion Token Service Domain...Done
Creating ptokenresource Realm...Done
Creating vtokenresource Realm...Done
Creating uatokenresource Realm...Done
Creating ptokenresource GET Rule...Done
Creating vtokenresource GET Rule...Done
Creating uatokenresource GET Rule...Done
Creating CTS Policy...Adding rules to CTS Policy...Done

10. The installation is complete:

The configuration is now complete. Press the Enter key to exit.

Update the CTS Domain for Your Environment

Once the base policies have been created, the objects can be updated to reflect your specific requirements. To update these policies:

1. Log into the SiteMinder Policy Server Administration Console
2. Navigate to the domain specified in step 5 above

View Domain: SDG Conversion Token Service Domain
[Domains](#) > View Domain: SDG Conversion Token Service Domain


General | Realms | Policies | Responses | Rule Groups | Variables

General

Name	SDG Conversion Token Service Domain	Description	SDG Conversion Token Service Domain
-------------	-------------------------------------	--------------------	-------------------------------------

Global Policies Apply ☒

User Directories

Name	Description
 Demo User Directory	Demo User Directory

3. Click on Policies:

Modify Domain: SDG Conversion Token Service Domain
[Domains](#) > Modify Domain: SDG Conversion Token Service Domain

General | Realms | **Policies** | Responses | Rule Groups | Variables

• = Required

Policies

Name	Description
 SDG CTS Policy	Map users to resources, define additional attributes and configure other CTS items 

Create

4. Modify SDG Conversion Token Service Policy:

Modify Policy: SDG Conversion Token Service Policy
Domains > Modify Domain: SDG Conversion Token Service Domain > Modify Policy: SDG Conversion Token Service

General **Users** Rules Expression

• = Required

User Directories

Demo User Directory

Allow Nested Groups ☐
AND Users/Groups ☐

Name	User Class	Exclude
No results.		

Add Members Add Entry Add All

5. Add the list of authorized CTS users to the Policy:

Modify Policy: SDG Conversion Token Service Policy
Domains > Modify Domain: SDG Conversion Token Service Domain > Modify Policy: SDG Conversion Token Service

General **Users** Rules Expression

• = Required

User Directories

Demo User Directory

Allow Nested Groups ☐
AND Users/Groups ☐

Name	User Class	Exclude
All	All	<input type="checkbox"/> Exclude

Add Members Add Entry Add All

6. Submit the Changes to save the update

Configure Additional Agent Config Object Settings

The CTS Agent Config Object (ACO) settings can further refine the behavior used by the CTS SiteMinder connector. The default ACO name for the CTS is `sdg_conversiontokenservices_aco`.

The following table lists the ACO settings used by the CTS SiteMinder connector:

Parameter	Default	Description
CookieDomain	N/A	Reserved for future use
CookieName	N/A	Reserved for future use
CookiePath	N/A	Reserved for future use
DefaultAgentName	<i>None</i>	Defines a name that the agent uses to process authorization requests. The value for DefaultAgentName is used for requests on an IP address or interface when no agent name value exists in the AgentName parameter.
IdentityAttribute	<i>None</i>	Header value to look for to override the ID passed to client.
AgentName	<i>None</i>	Used to map a submitted resource to agent name for authorization.
DefaultAction	GET	Default action to use when evaluating requests against the Policy Server (e.g. GET POST PUT).
password	/ptokenresource	Default resource to use when evaluating requests against the Policy Server to map password tokens.
ServiceAgentName	sdg_conversion-tokenservices_wa	Agent name to use for default service requests to the Policy Server.
userAttributes	/uatokenresource	Default resource to use when evaluating requests against the Policy Server to map userAttributes tokens.
ValidationResource	/vtokenresource	Default resource to use when validating SiteMinder session tokens.

CA SiteMinder Java SDK and Policy Server Process

As stated earlier we include all the necessary files from CA in order to do what is needed for Host Registration and the JARS needed.

Prior to completing this step please fill out the following table to have the needed pieces of information to complete the host registration.

SiteMinder Policy Server IP Address	
SiteMinder Administrator User (Or User with Agent Registration Permission)	
SiteMinder User Password	
Trusted Host Name for CTS	
SiteMinder HCO (Host Config Object)	

Register the SDK Agent Used by the SDG Conversion Token Service

Below are the steps to do the host registration procedure using the files we provided in the CTS deployment kit you should've downloaded already. The run location is the directory you noted above where you extracted all the files. (*The instructions are going to assume Linux and a base directory of opt/CTS_Install*)

1. In /opt/CTS_Install/CTS_2212_Package/ca-sdk, vi the smreghost.sh or smreghost.bat if on windows to add the required SDK Files Path:

```
#!/bin/ksh

#####
#####
###
## Copyright (c) 2006 CA. All rights reserved.          ##
## This software may not be duplicated, disclosed or reproduced in whole or      ##
## in part for any purpose except as authorized by the applicable license agreement, ##
## without the express written authorization of CA. All authorized reproductions   ##
## must be marked with this language.          ##
##                                     ##
## TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS          ##
## SOFTWARE "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING ##
## WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY,          ##
## FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT          ##
## WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS          ##
## OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS MATERIAL,          ##
## INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS          ##
## INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY          ##
## ADVISED OF SUCH LOSS OR DAMAGE.          ##
#####
#####
###

export JAVA_HOME=Your Java Home
export SM_SMREGHOST_CLASSPATH=/opt/CTS_Install/CTS_2212_Package/ca-sdk/java64
/smagentapi.jar: /opt/CTS_Install/CTS_2212_Package/ca-sdk/java64 /cryptoj.jar export
PATH=$JAVA_HOME/bin:$PATH

java -classpath "$SM_SMREGHOST_CLASSPATH" com.ca.siteminder.sdk.agentapi.SmRegHost "$@"
# The caller needs the exit status from SmRegHost exit $?
```

2. In the same folder where you edited the smregghost.bat (Windows) or smregghost.sh (Linux) run the following command respective of you OS and with the following syntax:
 - a. Windows: smregghost.bat -i ps_ip_addr -u sm_admin -p sm_admin_pw -hn trusted_host_name -hc sm_hco
 - b. Linux: ./smregghost.sh -i ps_ip_addr -u sm_admin -p sm_admin_pw -hn trusted_host_name -hc sm_hco
 - c. Example (Linux): ./smregghost.sh -i 192.168.1.225 -u siteminder -p p@ss1234! -hn cts_pf -hc cts_pf_hco
3. Remember the location of this directory and file as you will need to copy later. (ie. /opt/CTS_Install/CTS_2212_Package/cs-sdk/SmHost.conf)

Certificate & KeyStore for Mutual Authentication

Generate a Certificate KeyStore for mutual authentication. Details on how to configure the certificate and keystore are detailed in the following section.

Import client certificate to be used into KeyStore.

NOTE: The install comes with a pre-configured sample keystore on the client and server side. This can be used as a demo if needed without any further configuration.

SDG Conversion Token Service Configuration Parameters

By default, nothing needs to be changed if the steps above are followed.

cts.properties File

The following table outlines the CTS configuration parameters:

Parameter	Description	Example
agentSmHostConfigFile	Absolute or relative location of the cbSmHost.conf file	config/cbSmHost.conf
agentConfigObject	Agent Configuration Object to be used	sdg_conversiontokenservices_aco
authenticatedUserSerialNumberPath	Absolute or relative location of the text file containing the list of certificate serial numbers to be used in mutual authentication	config/authenticatedUsers.txt
authRequestURI	This is the string that is used for mutual authentication. Do not change this value	/tokenservicesimulator/1/token/

jetty.xml File

The jetty.xml file under the config directory contains the Jetty server configuration.

The following table outlines the critical configuration parameters:

Parameter	Description	Example
securePort	Server port to use when connecting to the Jetty server.	<code><Set name="securePort"><Property name="jetty.ssl.port" default="8443" />8586</Set></code>
keyStorePath	Location of the keystore containing the required SSL certificates.	<code><Set name="keyStorePath"> config/cts_server.keystore</Set></code>
keyStorePassword	Key Store password to use when accessing the Key Store	<code><Set name="keyStorePassword"> keystorepassword</Set></code>
keyManagerPassword	The password for the imported key	<code><Set name="keyManagerPassword"> keystorepassword</Set></code>
trustStorePath	Location of the trustStore containing the required SSL certificates. Can be the same as the keystore	<code><Set name="trustStorePath"> config/cts_server.keystore</Set></code>
trustStorePassword	Trust Store password to use when accessing the Key Store	<code><Set name="trustStorePassword"> keystorepassword</Set></code>

The file should be modified based upon the imported key and certificate.

CHAPTER 3: STARTING THE CTS SERVICE

The included *run.bat* or *run.sh* files start the CTS. The files are located in the <CTS HOME> directory.\

The files can be run directly or through the command line.

CHAPTER 4: CONFIGURING MUTUAL CLIENT CERTIFICATE AUTHENTICATION

To configure the SDG Conversion Token Service for mutual SSL, certificates must be configured both on the client and server side to ensure that the client is authorized to request tokens and also for it to be able to connect securely to the CTS.

The following tools are used to create the and import the certificates in this section:

- Java keytool
- openssl

Other tools can be used for this purpose. However, those tools are not documented in this guide. The keytool can be found at <JDK HOME>\bin or <JDK HOME>/bin depending on the Operating System.

The following steps are leveraged to configure mutual SSL:

Client Side:

1. Generate a client certificate key pair
2. Export client certificate in .cer format
3. Export client certificate in .pem format
4. Export client certificate private key in .pem format

Server Side:

1. Generate a keystore keypair
2. Import client certificate into keystore as a trusted certificate
3. Obtain certificate serial number from client certificate and add to authenticateusers.txt file

Client Side Certificate Generation

Generate a Client Certificate Key Pair

Keytool can be used to generate the client private key and certificate. Keytool is run from the command prompt.

The format for generating the certificate with keytool is as follows:

```
keytool -genkeypair -alias <ALIAS NAME> -keystore <KEY STORE NAME> -storetype pkcs12  
keyalg RSA
```

For example:

```
keytool -genkeypair -alias client -keystore client.p12 -storetype pkcs12 -keyalg RSA
```

After running the command, you will be prompted for several values. These values are specific to your environment. For example:

```
> keytool -genkeypair -alias client -keystore client.p12 -storetype pkcs12 -keyalg RSA
> Enter keystore password: keystorepassword >
Re-enter new password: keystorepassword >
What is your first and last name?
    [Unknown]: Client Certificate
> What is the name of your organizational unit?
[Unknown]: Client Services > What is the name
of your organization? [Unknown]: sdgc.com
> What is the name of your City or Locality? [Unknown]: New
York > What is the name of your State or Province? [Unknown]:
NY > What is the two-letter country code for this unit?
[Unknown]: US
> Is CN=Client Certificate, OU=Client Services, O=sdgc.com, L=New York, ST=NY, C=US correct?
    [no]: yes
```

The result of this example will generate the following file: client.p12

Export client certificate without Private Key for use on server side (.cer file)

The client certificate must then be imported into the Jetty server so that the CTS can validate the client is authorized to make token requests. Keytool is run from the command prompt.

Export the Certificate for Use on the CTS Server

The format for exporting the certificate using keytool is as follows:

```
keytool -exportcert -alias <CERTIFICATE ALIAS> -file <CERTIFICATE FILE NAME> -keystore
<KEY STORE NAME> -storetype pkcs12 -storepass <KEY STORE PASSWORD>
```

So, for our example, the command is:

```
keytool -exportcert -alias client -file client_cert.cer -keystore client.p12 -storetype
pkcs12 -storepass keystorepassword
```

The following file is generated for this example: client_cert.cer

Copy this file to the server where the SDG Conversion Token Service is being executed. This certificate will be used as a trusted certificate in the keystore used by the SDG Conversion Token Service.

Convert the Certificate to pem Format

For our example, openssl is used to convert the certificate to pem format. This certificate will be used later in the document with curl for testing the operation of the CTS. Openssl is run from the command line. On Windows, openssl can be downloaded. Other options also exist for obtaining openssl (e.g. cygwin).

The format for generating the pem certificate is as follows:

```
openssl x509 -inform der -in client_cert.cer -out client_cert.pem
```

So, for our example, the command is:

```
openssl x509 -inform der -in client_cert.cer -out client_cert.pem
```

The following file is generated: client_cert.pem

Extract the Private Key

For our example, openssl is used to extract the private key. This key will be used later in the document with curl for testing the operation of the CTS. openssl is run from the command line. On Windows, openssl can be downloaded. Other options also exist for obtaining openssl (e.g. cygwin).

The format for extracting the private key is as follows:

```
openssl pkcs12 -nodes -in <KEY STORE> -out <KEY FILE NAME>
```

After running the command, you will be prompted for the key store password.

So, for our example, the command is:

```
> openssl pkcs12 -nodes -in client.p12 -out clientkey.pem  
> Enter Import Password: keystorepassword MAC verified  
OK
```

The following file is generated: client_key.pem

Files Generated

The following files were generated for the documented example steps:

- client.p12
- client_cert.cer
- client_cert.pem
- client_key.pem

Server Side Certificate Generation and Mutual SSL Configuration

Generate a Keystore Key Pair

Keytool is used to configure SSL for the Jetty server. This allows clients to connect to the CTS over SSL to ensure that the communication is encrypted. On the server where the SDG Conversion Token Service is to be executed, go to the <CTS HOME>\config or <CTS HOME>/config directory depending on the Operating System.

The format for generating the certificate with keytool is as follows:

```
keytool -genkey -alias <ALIAS> -keyalg RSA -keyStore <KEY STORE NAME> -keysize 2048 sigalg "SHA1withRSA"
```

For example:

```
keytool -genkey -alias cts -keyalg RSA -keyStore cts_server.keystore -keysize 2048 sigalg  
"SHA1withRSA"
```

After running the command, you will be prompted for several values. These values are specific to your environment. For example:

```
> keytool -genkey -alias cts -keyalg RSA -keyStore cts_server.keystore -keysize 2048  
sigalg "SHA1withRSA"  
> Enter keystore password: keystorepassword >  
Re-enter new password: keystorepassword >  
What is your first and last name?  
[Unknown]: SDG Conversion Token Service > What  
is the name of your organizational unit?  
[Unknown]: Services  
> What is the name of your organization?  
[Unknown]: sdgc.com  
> What is the name of your City or Locality? [Unknown]: New  
York  
> What is the name of your State or Province?
```

```
[Unknown]: NY > What is the two-letter country code for
this unit? [Unknown]: US
> Is CN=SDG Conversion Token Service, OU=Services, O=sdgc.com, L=New York, ST=NY, C=US
correct?
[no]: yes
> Enter key password for <cts>
(RETURN if same as keystore password): <press RETURN>
```

The following file is generated: cts_server.keystore

Copy the generated key store to the <CTS HOME>\config or <CTS HOME>/config directory depending on the Operating System. If another location is used, the jetty.xml file must be updated to reflect the new location.

Copy the Client Certificate to the Server

In order validate the client connecting to the CTS, the certificate generated in the client certificate section above. Copy the client certificate into the <CTS HOME>\config or <CTS HOME>/config directory depending on the Operating System. For the example in this document, copy the client_cert.cer file generated in the previous steps into the config directory.

Import the Client Certificate into the keystore as a Trusted Certificate

The client certificate must be imported into the keystore so that the client can be validated during calls to the CTS. Keytool is run from the command line.

The format for importing the certificate is as follows:

```
keytool -importcert -keystore <KEY STORE NAME> -alias <ALIAS> -file <CERTIFICATE FILE> -v
-trustcacerts -noprompt -storepass <KEY STORE PASSWORD>
```

After running the command, you will be prompted for the key store password.

So, for our example, the command is:

```
> keytool -importcert -keystore cts_server.keystore -alias client -file client_cert.cer v
-trustcacerts -noprompt -storepass keystorepassword > Certificate was added to keystore
[Storing cts_server.keystore]
```

Verify the Certificate in the Keystore

Once the certificate is imported, the key store contents should be validated to ensure that the certificate imported correctly. This step is also used for obtaining the client certificate serial number.

This serial number is added to the `authenticatedusers.txt` file. The file contains the list of client certificates allowed to query the CTS. Keytool is run from the command line.

The format for validating the certificate is as follows:

```
keytool -v -list -keystore <KEY STORE NAME> -storepass <KEY STORE PASSWORD>
```

So, for our example, the command is:

```
> keytool -v -list -keystore cts_server.keystore -storepass keystorepassword
```

Keystore type: JKS

Keystore provider: SUN

Your keystore contains 2 entries

Alias name: client

Creation date: Apr 30, 2013

Entry type: trustedCertEntry

Owner: CN=Client Certificate, OU=Client Services, O=sdgc.com, L=New York, ST=NY, C=US

Issuer: CN=Client Certificate, OU=Client Services, O=sdgc.com, L=New York, ST=NY, C=US

Serial number: 51802ddb

Valid from: Tue Apr 30 13:47:23 PDT 2013 until: Mon Jul 29 13:47:23 PDT 2013 Certificate

fingerprints:

MD5: E2:D2:31:B8:FE:87:AE:5F:41:94:FF:DD:73:1D:8F:35

SHA1: BC:F6:09:3A:E6:DE:42:A7:C4:30:3C:A0:98:59:74:CB:2F:4B:EO:ED

Signature algorithm name: SHA1withRSA

Version: 3

```
Alias name: cts
Creation date: Apr 30, 2013
Entry type: PrivateKeyEntry
Certificate chain length: 1 Certificate[1]:
Owner: CN=SDG Conversion Token Service, OU=Services, O=sdgc.com, L=New York, ST=NY, C=US
Issuer: CN=SDG Conversion Token Service, OU=Services, O=sdgc.com, L=New York, ST=NY, C=US
Serial number: 51802e94
Valid from: Tue Apr 30 13:50:28 PDT 2013 until: Mon Jul 29 13:50:28 PDT 2013 Certificate
fingerprints:
    MD5: 81:8B:51:B7:11:3B:A2:45:7D:C6:99:01:FB:35:35:AB
    SHA1: F5:FA:EC:46:45:99:18:12:6C:DF:AA:EB:22:44:3E:01:49:F7:11:B2
    Signature algorithm name: SHA1withRSA
    Version: 3
```

```
*****
*****
```

For this example, the serial number required is bolded in the response above.

Add Client Certificate Serial Number to the `authenticateusers.txt` File

The client certificate serial number is added to the `authenticatedusers.txt` file. The file contains the list of client certificates allowed to query the CTS. Copy the serial number from the client certificate alias. Open the `authenticateusers.txt` file and add the value into the file.

For the documented example, the value is: `51802ddb`

About SDG

With more than 30 years of experience partnering with global enterprises on complex business and IT initiatives, SDG is a trusted provider of advisory, transformation, and managed services. The firm empowers organizations to strengthen cyber resilience by integrating AI into identity, threat, and risk management solutions that protect digital assets and deliver measurable business value. Learn more at www.sdgc.com.



- 75 North Water Street Norwalk, CT 06854
- 203.866.8886
- sdgc.com