



## **COREBLOX TOKEN SERVICE**

### **2.2.x INSTALLATION/CONFIGURATION - EMBEDDED/STANDALONE**

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 CoreBlox LLC 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 CoreBlox. This Documentation is confidential and proprietary information of CoreBlox 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 CoreBlox governing your use of the CoreBlox software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CoreBlox.

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 CoreBlox 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 CoreBlox that all copies and partial copies of the Documentation have been returned to CoreBlox or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, COREBLOX LLC 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 COREBLOX 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 COREBLOX 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 CoreBlox LLC.

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 © 2020 CoreBlox LLC. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

## Contacting CoreBlox

**Contact Information:**

420 Lexington Avenue, Suite 455  
New York, NY 10170

Toll Free: 1-877-879-2569 (1-877-TRY-BLOX)

Direct: 1-617-275-7860

Fax: 1-508-405-2273

Email: [info@coreblox.com](mailto:info@coreblox.com)

Web: [www.coreblox.com](http://www.coreblox.com)

**Community:**

CoreBlox Blog: <http://www.coreblox.com/blog>

## Contents

Chapter 1: Overview .....	3
CoreBlox Token Service Overview .....	3
Workflow Integration with Ping Federate .....	3
Platform Support .....	4
Components of the CoreBlox Token Service .....	4
Component Location .....	5
Which Install Path Should You Choose? .....	5
Chapter 2: CoreBlox Token Service PingFederate Embedded Install .....	6
Download Components .....	6
Communication Port Requirements .....	6
Stage Installation to Dedicated Folder .....	7
CA SiteMinder Java SDK and Policy Server Process .....	7
Register the SDK Agent Used by the CoreBlox Token Service .....	7
Create the SiteMinder Policy Objects for CTS.....	8
SiteMinder Policy Base Object Creation .....	9
Update the CTS Domain for Your Environment.....	10
PingFederate.....	12
Generate A Client Certificate for CoreBlox Token Service .....	12
CoreBlox Token Service .....	15
JAVA .....	15
Installation of the CoreBlox Token Service WAR file.....	15
Configuration of the CoreBlox Token Service .....	16
Copy and Update the ‘authenticatedUsers.txt’ file.....	16
Copy and Rename the SmHost.conf to cbSmHost.conf .....	17
Copy the CTS Integration Kit JAR File to PingFederate.....	17
Copy the CA SiteMinder Java SDK JAR Files to PingFederate.....	17
Edit the run.properties.....	17
Edit ‘cts.properties’ file .....	18
Restart PingFederate for CoreBlox Token Service to Deploy.....	18
Validate the CoreBlox Token Service .....	18
Testing the CoreBlox Token Service.....	19
Deploying CoreBlox Token Service to Additional Nodes .....	19
Chapter 3: CoreBlox Token Service Standalone Install for SiteMinder .....	19
Install Package Overview .....	20
CTS Directory Structure .....	20
CoreBlox Token Service File Details .....	21
Creation of the CA SiteMinder Policy Server Objects .....	23
SiteMinder Policy Base Object Creation.....	24
Update the CTS Domain for Your Environment .....	25
Configure Additional Agent Config Object Settings .....	27
CA SiteMinder Java SDK and Policy Server Process .....	28
Register the SDK Agent Used by the CoreBlox Token Service .....	28
Certificate & KeyStore for Mutual Authentication .....	29

CoreBlox Token Service Configuration Parameters .....	29
cts.properties File.....	29
jetty.xml File.....	30
Chapter 4: Starting the CTS Service.....	31
Chapter 5: Configuring Mutual Client Certificate Authentication .....	32
Client Side Certificate Generation .....	32
Generate a Client Certificate Key Pair .....	32
Export client certificate without Private Key for use on server side (.cer file) .....	33
Export the Certificate for Use on the CTS Server .....	33
Convert the Certificate to pem Format .....	33
Extract the Private Key .....	34
Files Generated .....	34
Server Side Certificate Generation and Mutual SSL Configuration.....	35
Generate a Keystore Key Pair.....	35
Copy the Client Certificate to the Server.....	35
Import the Client Certificate into the keystore as a Trusted Certificate .....	36
Verify the Certificate in the Keystore .....	36
Add Client Certificate Serial Number to the authenticateusers.txt File .....	37

## Chapter 1: Overview

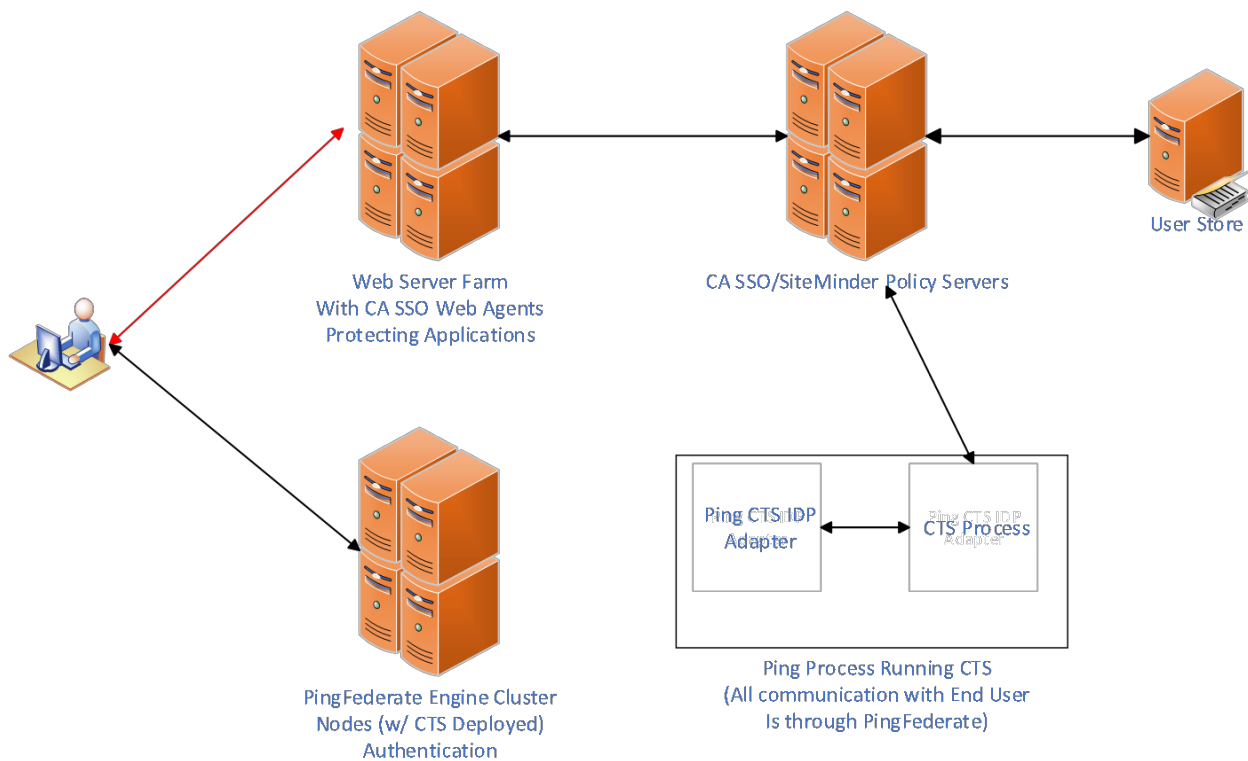
### CoreBlox Token Service Overview

The CoreBlox 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.

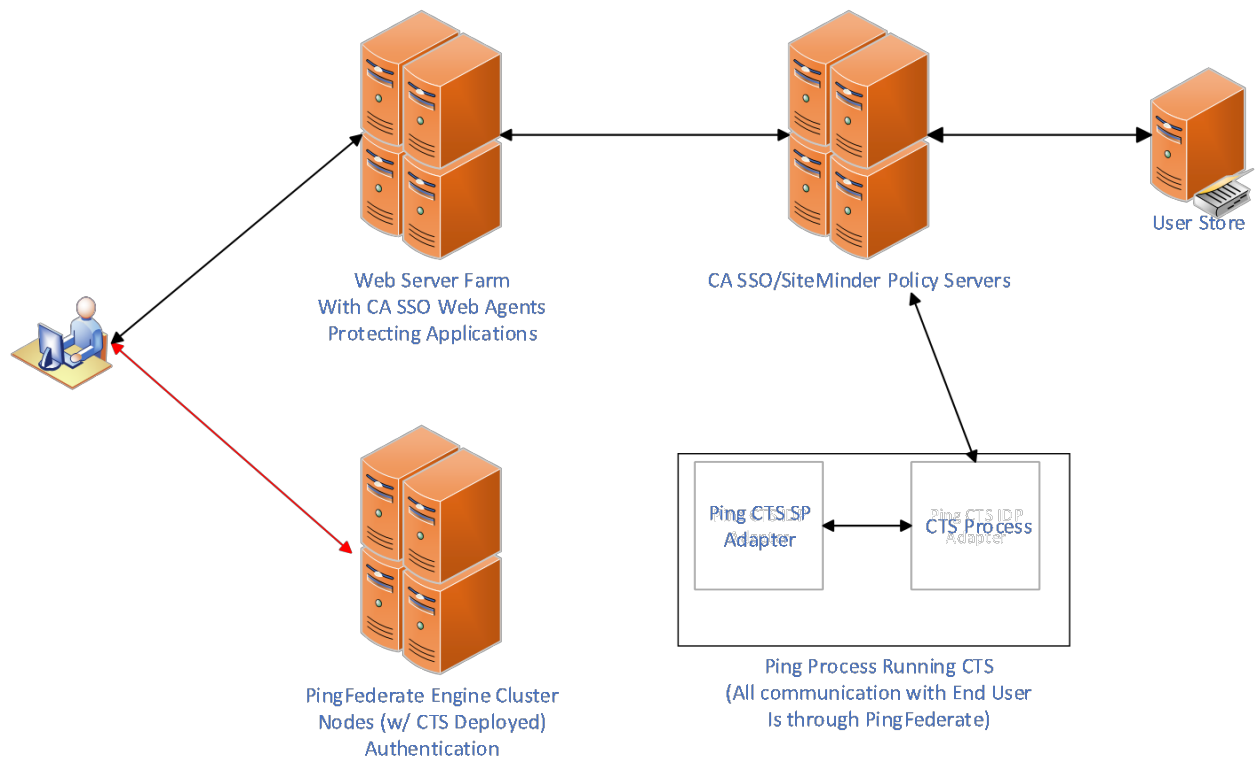
It is built with the CA SSO SDK that allows it to interact with CA SiteMinder Policy Servers. It supports a level of communication that a full featured web agent leverages to talk to multiple policy servers and supports round robin and failover communication.

### Workflow Integration with Ping Federate

The CoreBlox 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 Coreblox Token Adapters to implement the following use cases.



**Use Case 1 – User authenticates to a CA SSO/SSO Resource,** existing or newly created SMOSESSION Tokens can then be exchanged for access to a Ping Federate protected application with no additional challenge through the use of the CoreBlox 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 CoreBlox 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"> <li>PingFederate 9.x</li> <li>PingFederate 10.x</li> </ul>	<ul style="list-style-type: none"> <li>SiteMinder 12.52</li> <li>SiteMinder 12.6</li> <li>SiteMinder 12.7</li> <li>SiteMinder 12.8</li> </ul>	<ul style="list-style-type: none"> <li>Oracle Java 1.8</li> <li>Oracle Java 11*</li> <li>Amazon Corretto 11*</li> <li>OpenJDK 11*</li> </ul>

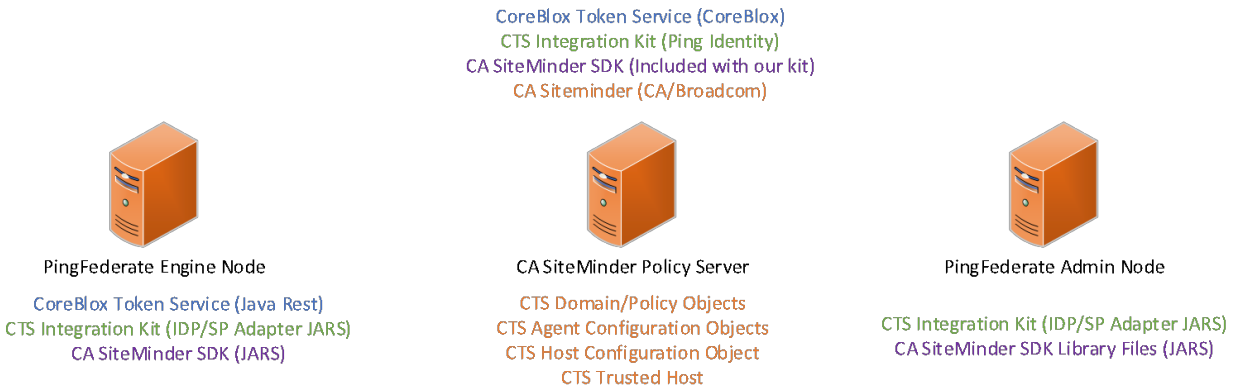
\* The SiteMinder Agent SDK is only supported on Oracle 1.8, Oracle (formerly Sun) 1.7, and IBM Java 1.7.x & 1.8.x. CoreBlox cannot guarantee support of other versions of Java since they are unsupported by Broadcom.

## Components of the CoreBlox Token Service

The following components are required as part of the collective solution

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

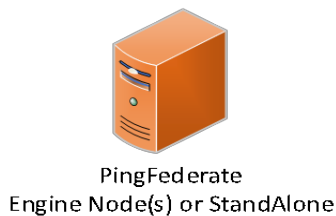
## CoreBlox Token Service Ping Federate Installation – Version 2.2.x



### Component Location

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

PingFederate	CoreBlox 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



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

### Which Install Path Should You Choose?

The CoreBlox 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: CoreBlox Token Service PingFederate Embedded Install

Let's begin with what is needed to setup CTS with PingFederate and get it up and running.

### Download Components

This document assumes you already have CA SiteMinder installed and configured as per CA/Broadcom documentation. It also assumes that PingFederate has also been installed and baseline configuration done to have it up and running ready to start. In order to complete the solution the following components need to be downloaded:

- CoreBlox Token Service 2.2.x
  - <https://www.coreblox.com/products/coreblox-token-service>
  - **Please note a license key for CTS will be emailed to the address provided for download within 1 business day. If you have not received your license key within that time please send an email to [support@coreblox.com](mailto:support@coreblox.com).**
- CTS Integration Kit (latest version)
  - <https://www.pingidentity.com/en/resources/downloads/pingfederate.html>
    - Click on the Add-ons section to see the integration kits, locate the **CoreBlox Integration Kit 2.6.2, as of 08/02/2020.**

### Communication Port Requirements

The following table breaks down the Communication Ports the solution requires in order to fully function.

PingFederate Engine Node(s)

Port	CTS Adapter
<b>9032 (Inbound)</b>	PingFederate installs with a default Port of 9031 so 9032 assumes you would just make that the second port. If you wish to use another port that is fine, the requirement is the <b>HTTP.SECONDARY PORT</b> must be enabled and set.
<b>44441 (Outbound)</b>	CA SiteMinder Accounting Port
<b>44442 (Outbound)</b>	CA SiteMinder Authentication Port
<b>44443 (Outbound)</b>	CA SiteMinder Authorization Port

PingFederate Administrator Console

Port	CTS Adapter
<b>9032 (Inbound)</b>	PingFederate installs with a default Port of 9031 so 9032 assumes you would just make that the second port. If you wish to use another port that is fine, the requirement is the <b>HTTP.SECONDARY PORT</b> must be enabled and set. (Whatever port you specify on the Engine Nodes has to match the Admin Node)

### Stage Installation to Dedicated Folder

It is recommended to build a folder where you will expand the CTS files and use that as the base folder for running all commands and configurations. It is best that this folder be located outside of the PingFederate Installation directory but on the same server where PingFederate is located. **Note: This folder will be used frequently during the installation process.**

Example:

**Windows:** d:\CTS\_Install

**Linux:** /opt/CTS\_Install or /apps/CTS\_Install

<b>Your CTS Location</b>	
--------------------------	--

### 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 CoreBlox 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 smreghost.bat (Windows) or smreghost.sh (Linux) run the following command respective of you OS and with the following syntax:
  - a. Windows: smreghost.bat -i ps\_ip\_addr -u sm\_admin -p sm\_admin\_pw -hn trusted\_host\_name -hc sm\_hco
  - b. Linux: ./smregshost.sh -i ps\_ip\_addr -u sm\_admin -p sm\_admin\_pw -hn trusted\_host\_name -hc sm\_hco
  - c. Example (Linux): ./smregshost.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)

## Create the SiteMinder Policy Objects for CTS

The CoreBlox Token Service requires a set of policies to exist in the environment. In addition to the configuration steps outlined here, a User Directory is required. This section assumes these items already exist.

There are a couple of 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):

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:
6. An existing user directory is configured for authentication of user tokens. Specify the user directory to authenticate users that will be using CTS in the Domain Object User Directories.
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:
8. Confirm the installation parameters:

\*\*\*\*\* CoreBlox Token Service Policy Server Installer \*\*\*\*\*

Enter Policy Server Administrator credentials -----

Administrator ID: siteminder

Administrator Password: <siteminder password>

Enter unique CTS Domain Name -----

Name (or X to exit): **CoreBlox Token Service Domain**

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**

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}**)

Confirm Installation Parameters -----

[1] CTS Domain Name: CoreBlox 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:

CoreBlox Token Service Install and Configuration Guide – Version 1.0 Build V3

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 - CoreBlox Token Service Domain .. Done

Creating ptokenresource Realm.... Done  
Creating vtokenresource Realm .... Done  
Creating uatokenresource Realm .. Done

Creating ptokentresource GET Rule ... Done  
Creating vtokentresource GET Rule.... Done  
Creating uatokentresource GET Rule .. Done  
Creating CTS Policy...Adding rules to CTS Policy....Done

10. The installation is complete.

## Update the CTS Domain for Your Environment

Once the base policies have been created, the objects can be updated to reflects 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

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

**View Domain: CoreBlox Token Service Domain**  
[Domains](#) > View Domain: CoreBlox Token Service Domain


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

**General**

**Name** CoreBlox Token Service Domain **Description** CoreBlox Token Services Configuration Domain

**Global Policies Apply** ☒

**User Directories**

Name	Description
 Demo User Directory	Demo User Directory



### 3. Click on Policies

**Modify Domain: CoreBlox Token Service Domain**  
[Domains](#) > Modify Domain: CoreBlox Token Service Domain

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

• = Required

**Policies**

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

### Modify CoreBlox Token Service Policy

4. Add the list of authorized CTS users to the Policy
5. Submit the Changes to save the update

**Modify Policy: CoreBlox Token Service Policy**Domains > [Modify Domain: CoreBlox Token Service Domain](#) > Modify Policy: CoreBlox Token Service Policy

<b>General</b>	<b>Users</b>	<b>Rules</b>	<b>Expression</b>
----------------	--------------	--------------	-------------------

• = Required

**User Directories**

**Demo User Directory**

Allow Nested Groups ☐  
 AND Users/Groups ☐

Name	User Class	Exclude
No results.		

Add Members Add Entry Add All

**Modify Policy: CoreBlox Token Service Policy**Domains > [Modify Domain: CoreBlox Token Service Domain](#) > Modify Policy: CoreBlox Token Service Policy

<b>General</b>	<b>Users</b>	<b>Rules</b>	<b>Expression</b>
----------------	--------------	--------------	-------------------

• = 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

This completes the configuration of the needed Policies, Objects, and Rules in CA SiteMinder. We now can go back to PingFederate and complete the steps there to get the CTS Service up and talking.

**PingFederate**

One of the core pieces that needs to be completed is creating a self-signed certificate the CTS uses to talk securely to PingFederate and swap tokens.

**Generate A Client Certificate for CoreBlox Token Service**

From the PingFederate Administrative Console Select Security -> SSL Client Keys and Certificates

- 1) Create New

Create a new Certificate and Private Key.

COMMON NAME	CoreBlox CTS
ORGANIZATION	CoreBlox LLC
ORGANIZATIONAL UNIT	
CITY	
STATE	
COUNTRY	US
VALIDITY (DAYS)	365
KEY ALGORITHM	RSA
KEY SIZE (BITS)	2048
SIGNATURE ALGORITHM	RSA SHA256

Click “Next” and “Done”

- 2) Record the serial number of the certificate

<b>CTS Client Certificate Serial Number</b> (Enter with no :’s) <b>EX: 0167BED9C9CA</b>	
---	--

- 3) Click Export to Save the certificate to your desktop

01:67:BE:D9:C9:CA	CN=CoreBlox CTS, O=CoreBlox LLC, C=US	Tue Dec 17 17:06:43 PST 2019	RSA 2048	Valid	<a href="#">Export</a> <a href="#">Certificate Signing</a> - Certificate not saved <a href="#">Delete</a>
-------------------	--	------------------------------	----------	-------	---

- 4) Select “Certificate Only” and click “Next”

Certificate Management | Export Certificate

Export Certificate    Export & Summary

You have a choice of exporting the certificate and the key or just the certificate.

☒ CERTIFICATE ONLY  
☐ CERTIFICATE AND PRIVATE KEY

Cancel    [Next](#)

- 5) Click “Export” and “Done”



Certificate Management | Export Certificate

Export Certificate   Export & Summary

Click the Export button to export this certificate to the file system.

Export Certificate	
Subject DN	CN=CoreBlox CTS, O=CoreBlox LLC, C=US
Issuer DN	CN=CoreBlox CTS, O=CoreBlox LLC, C=US
Serial Number	01:67:BE:D9:C9:CA
Expires	Tue Dec 17 17:06:43 PST 2019

Export

Cancel   Previous   Done

- 6) From the PingFederate Administrative Console -> Security -> Trusted CA's select Import and choose the exported certificate you saved to your desktop in the previous step and click "Next"

Certificate Management | Import Certificate

Import Certificate   Summary

Please select the file containing the desired certificate.

FILENAME   167BED9C9CA.crt   Choose file

Cancel   Next

- 7) Verify the certificate and click "Save"

Certificate Management | Import Certificate

Import Certificate | Summary

Summary

Import Certificate	
Filename	167BED9C9CA.crt
File Size	1084
Serial Number	01:67:BE:D9:C9:CA
Subject DN	CN=CoreBlox CTS, O=CoreBlox LLC, C=US
Issuer DN	CN=CoreBlox CTS, O=CoreBlox LLC, C=US
Expires	Tue Dec 17 17:06:43 PST 2019
Signature Algorithm	SHA256withRSA
MD5 Fingerprint	50E5DEC8BC94781F7B75CB294F2FC839
SHA1 Fingerprint	7DEB2F5D3D8C7842FACCD165387C991FC4435E7E

Cancel Previous Done **Save**

## CoreBlox Token Service

In this section we will deploy the CoreBlox Token Service WAR file to the PingFederate Node, these steps should be performed as the user that owns and will run the PingFederate process.

### JAVA

Ensure your JAVA\_HOME Environment variables are set before proceeding. In Windows these are set in your system environment variables, in Linux these should be set in the appropriate *shell\_profile.sh* script.

### Installation of the CoreBlox Token Service WAR file

- 1) From your PingFederate engine node folder navigate to the following folder
  - a. .../pingfederate\_home/pingfederate/server/default/deploy
  - b. Create a directory called "coreblox-tokenservice.war"
  - c. Example: /opt/PingFederate.10.1.0/pingfederate/server/default/deploy/coreblox-tokenservice.war
- 2) Change into the newly created directory in the previous step
  - a. `cd /opt/PingFederate.10.1.0/pingfederate/server/default/deploy/coreblox-tokenservice.war` (Make sure to use your path this is an example)
- 3) Run the following command `'jar -xvf /opt/CTS_Install/CTS_2212_Package/coreblox-tokenservice-combined-2.2.12.war`
- 4) Verify the directory structure matches the sample below

```

drwxr-xr-x. 10 pfuser pfadmin 4096 Nov  6 09:03 ..
drwxr-x---.  4 pfuser pfadmin 4096 Nov  6 09:03 META-INF
drwxr-x---. 10 pfuser pfadmin 4096 Nov  6 09:03 com
-rw-r-----.  1 pfuser pfadmin  938 Nov  6 09:03 log4j.properties
-rw-r-----.  1 pfuser pfadmin 1259 Nov  6 09:03 Launch.class
drwxr-x---.  4 pfuser pfadmin 4096 Nov  6 09:03 WEB-INF
drwxr-x---.  3 pfuser pfadmin   25 Nov  6 09:03 io
drwxr-x---.  2 pfuser pfadmin   32 Nov  6 09:03 groverconfig8491016507689653801
drwxr-x---. 11 pfuser pfadmin 4096 Nov  6 09:03 org
drwxr-x---.  2 pfuser pfadmin   35 Nov  6 09:03 mozilla
drwxr-x---.  5 pfuser pfadmin   46 Nov  6 09:03 javax
-rw-r-----.  1 pfuser pfadmin 2052 Nov  6 09:03 about.html
-rw-r-----.  1 pfuser pfadmin  319 Nov  6 09:03 jetty-dir.css
drwxr-x---.  3 pfuser pfadmin   23 Nov  6 09:03 netegrity
-rw-r-----.  1 pfuser pfadmin 1073 Nov  6 09:03 cryptoPerms
-rw-r-----.  1 pfuser pfadmin  740 Nov  6 09:03 smagentapibuild.properties
drwxr-x---.  3 pfuser pfadmin   23 Nov  6 09:03 jersey
drwxr-x---.  4 pfuser pfadmin   28 Nov  6 09:03 afu
drwxr-x---. 13 pfuser pfadmin 4096 Dec 18 05:47 .

```

- 5) Copy the “license.lic” file that you received from CoreBlox to the same directory you are in. The directory structure should now include the license file as shown below:

```

-bash-4.2$ ls -lart
total 52
drwxr-xr-x. 10 pfuser pfadmin 4096 Nov  6 09:03 ..
drwxr-x---.  4 pfuser pfadmin 4096 Nov  6 09:03 META-INF
drwxr-x---. 10 pfuser pfadmin 4096 Nov  6 09:03 com
-rw-r-----.  1 pfuser pfadmin  938 Nov  6 09:03 log4j.properties
-rw-r-----.  1 pfuser pfadmin 1259 Nov  6 09:03 Launch.class
drwxr-x---.  4 pfuser pfadmin 4096 Nov  6 09:03 WEB-INF
drwxr-x---.  3 pfuser pfadmin   25 Nov  6 09:03 io
drwxr-x---.  2 pfuser pfadmin   32 Nov  6 09:03 groverconfig8491016507689653801
drwxr-x---. 11 pfuser pfadmin 4096 Nov  6 09:03 org
drwxr-x---.  2 pfuser pfadmin   35 Nov  6 09:03 mozilla
drwxr-x---.  5 pfuser pfadmin   46 Nov  6 09:03 javax
-rw-r-----.  1 pfuser pfadmin 2052 Nov  6 09:03 about.html
-rw-r-----.  1 pfuser pfadmin  319 Nov  6 09:03 jetty-dir.css
drwxr-x---.  3 pfuser pfadmin   23 Nov  6 09:03 netegrity
-rw-r-----.  1 pfuser pfadmin 1073 Nov  6 09:03 cryptoPerms
-rw-r-----.  1 pfuser pfadmin  740 Nov  6 09:03 smagentapibuild.properties
drwxr-x---.  3 pfuser pfadmin   23 Nov  6 09:03 jersey
drwxr-x---.  4 pfuser pfadmin   28 Nov  6 09:03 afu
-rw-r-----.  1 pfuser pfadmin  191 Nov  6 09:03 license.lic
drwxr-x---. 13 pfuser pfadmin 4096 Nov  6 09:03 .

```

## Configuration of the CoreBlox Token Service

The following sections will outline the steps necessary to complete the installation and configuration of the CoreBlox Token Service

### Copy and Update the ‘authenticatedUsers.txt’ file

- 1) Copy the ‘authenticatedUsers.txt’ file from ../pingfederate/server/default/deploy/coreblox-tokenservice.war/WEB-INF/ TO ../pingfederate/server/default/conf

- 2) Edit the 'authenticatedUsers.txt' file and enter the serial number of the certificate created earlier (Should be in a table on Page 13). Be sure to remove all ":" from the serial number.  
Example: "01:67:BE:D9:C9:CA" will be "0167BED9C9CA"



- 3) Save the file.

#### Copy and Rename the SmHost.conf to cbSmHost.conf

- 1) Copy the "SmHost.conf" file from /opt/CTS\_Install/CTS\_2212\_Package/ca-sdk/SmHost.conf TO  
../pingfederate/server/default/conf/cbSmHost.conf

#### Copy the CTS Integration Kit JAR File to PingFederate

- 1) From the folder where you downloaded the coreblox-integration-kit-2.6.2 file and extracted it, copy the coreblox-integration-kit-2.6.36.jar to the following location  
../pingfederate/server/default/deploy

#### Copy the CA SiteMinder Java SDK JAR Files to PingFederate

- 1) From the /opt/CTS\_Install/CTS\_2212\_Package/cs-sdk/java64 folder
  - a. cryptoj.jar
  - b. smagentapi.jar
- 2) Copy those files to ../pingfederate/server/default/lib

#### Edit the run.properties

- 1) Edit the 'run.properties' file located at ../pingfederate/bin/run.properties
- 2) Change the line pf.secondary.https.port=-1 to pf.secondary.https.port=9032
  - a. As stated earlier this is an example the port can be whatever you choose or need
- 3) Save the file

```

pf.https.port=9031

#
# This property defines a secondary HTTPS port that can be used, for example,
# with SOAP or artifact SAML bindings or for WS-Trust STS calls.
# To use this port, change the placeholder value to the port number
# you want to use.
# Important: If you are using mutual SSL/TLS for either WS-Trust STS
# authentication or for SAML back-channel authentication, you must use this
# port for security reasons (or use a similarly configured new listener,
# with either "WantClientAuth" or "NeedClientAuth" set to "true".
pf.secondary.https.port=9032

```

### Edit 'cts.properties' file

- 1) In the folder ../pingfederate/server/default/deploy/coreblox-tokenservice.war/WEB-INF/classes copy the '\_cts.properties' to 'cts.properties'
- 2) Edit the newly created 'cts.properties' to reflect the configuration, be sure to use exact path, whether using Windows or Linux all directories must be represented with a "/"
  - a. agentSmHostConfigFile is the path to the config directory where the cbSmhost.conf file was copied.
  - b. agentConfigObject is the name of the ACO created for CTS, this example below should be accurate if you used the perl script to create the policy store objects.
  - c. authenticatedUserSerialNumberPath is the path to the file 'authenticatedUsers.txt' file created to store the serial number of the certificate created earlier.
  - d. authResourceURI is the URL for CoreBlox, this should match the name of folder created under ../deploy with the .war removed
- 3) Your file should look like the example below:

```

agentSmHostConfigFile=/opt/pingidentity/pf/pingfederate/server/default/conf/cbSmHost.conf
agentConfigObject=coreblox_tokenservices_aco
authenticatedUserSerialNumberPath=/opt/pingidentity/pf/pingfederate/server/default/conf/authenticatedUsers.txt
authRequestURI=/coreblox-tokenservice/1/token/
forceCertValidation=no
useSharedSecret=no
altidnumber=0
usealtid=false
ignore_altid_characters=false
useTimeouts=true

```

- 4) Save File

### Restart PingFederate for CoreBlox Token Service to Deploy

- 1) Restart PingFederate in order to fully deploy the CoreBlox Token Service

### Validate the CoreBlox Token Service

- 1) Once PingFederate has restarted, check the server.log for the following line:

```
INFO
[org.eclipse.jetty.server.handler.ContextHandler] Started
o.e.j.w.WebAppContext@2bd02582{/coreblox-tokenservice-
2.0,file:///opt/pingidentity/pf/pingfederate/server/default/deploy/cor
eblox-tokenservice.war/,AVAILABLE}
```

- 2) Once you see this the CoreBlox Token Service has deployed successfully and should be up and ready to service requests.

### Testing the CoreBlox Token Service

To ensure the CoreBlox Token Service is up and ready to service requests you can run a test transaction through to validate it is ready and talking to SiteMinder correctly.

- 1) curl -k <https://localhost:9032/coreblox-tokenservice/1/token/test1234>
- 2) The Port used above in an example, if you used it fine, if not use whatever port you did setup PingFederate for the Secondary HTTPS Port.

If a 404 is returned the service may not have deployed or there may be a configuration issue

If you receive 'NO CONNECTION' then you know the CoreBlox Token Service is deployed, however it is having issues communicating with the SiteMinder Policy Servers.

If you receive 'INVALID TOKEN' then you have successfully deployed the CoreBlox Token Service and the connection to the CA SiteMinder policy servers was successful.

### Deploying CoreBlox Token Service to Additional Nodes

You may use the following steps to deploy the CoreBlox Token Service to additional clustered engine nodes

- 1) Copy the coreblox-tokenservice.war directory to the same location on the other nodes in the cluster as you deployed on the first one
- 2) Copy the authenticatedUsers.txt file to the same directory on the other nodes as deployed on the first one
- 3) Copy the cbSmHost.conf to the same location on the other nodes as is located on the first node
- 4) Open the cts.properties file on the node and make sure it is pointing to the correct path on the new nodes for each of those files. This file can also be copied to the other clustered engine nodes.
- 5) Copy the cryptoj.jar and smagentapi.jar files into the other nodes located at ../pingfederate/server/default/lib
- 6) Restart PingFederate on the new node
- 7) Repeat validation steps on each node as you complete all needed steps above.

## Chapter 3: CoreBlox Token Service 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 CoreBlox 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
coreblox-tokenservice-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 smreghost.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 3 <sup>rd</sup> 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.

### CoreBlox Token Service File Details

The following section lists the context of the example files used by the CTS.

Contents of run.sh:

```
JAVA_HOME=/apps/Java/jdk
CTS_HOME=/apps/CA/coreblox-tokenservice-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/coreblox-
tokenservice-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;coreblox-tokenservice-combined-2.0.war;. -
Dhttps.protocols=TLSv1.2 Launch config\jetty.xml
```

Example contents of authenticatedUsers.txt:

```
00C33C6C513105A65A
013B0A151EDA
```

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}2g1l/+2JdSiPhGHFxbnVrGAla2LaetSor8Chd0yUxoy19XKfaMABkokyPtP8cDk4F
fwNORZ4yCsFg6KmYUpR+mfgNXsG1kcwIMhJNvph2TmA/F4QI2G7PqyZ8ft79LmCYJqevtUt34IZxQpk
f0XR+4uRJjggEEHG0ZzRNsokKxz+08Vr22LT4500AoCEIOGO"
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>
      <Item>
        <New class="org.eclipse.jetty.server.HttpConnectionFactory">
          <Arg>
            <Ref refid="httpsConfig"></Ref>
          </Arg>
        </New>
      </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 CoreBlox 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):

```
***** CoreBlox 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): CoreBlox 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: CoreBlox 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 - CoreBlox Token Service Domain...Done  
 Creating ptokenresource Realm...Done  
 Creating vtokenresource Realm...Done  
 Creating uatokenresource Realm...Done  
 Creating ptokentresource GET Rule...Done  
 Creating vtokentresource GET Rule...Done  
 Creating uatokentresource 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: CoreBlox Token Service Domain**

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



General	Realms	Policies	Responses	Rule Groups	Variables
<b>General</b>					
<b>Name</b> CoreBlox Token Service Domain		<b>Description</b> CoreBlox Token Services Configuration Domain			
<b>Global Policies Apply</b> <input checked="" type="checkbox"/>					
<b>User Directories</b>					
<b>Name</b>		<b>Description</b>			
 Demo User Directory		Demo User Directory			

3. Click on Policies

**Modify Domain: CoreBlox Token Service Domain**[Domains](#) > [Modify Domain: CoreBlox Token Service Domain](#)

[General](#)
[Realms](#)
[Policies](#)
[Responses](#)
[Rule Groups](#)
[Variables](#)

• = Required

Policies	
Name	Description
 CoreBlox Token Service Policy	Map users to resources, define additional attributes and configure other CTS items 

[Create](#)

## 4. Modify CoreBlox Token Service Policy

**Modify Policy: CoreBlox Token Service Policy**[Domains](#) > [Modify Domain: CoreBlox Token Service Domain](#) > [Modify Policy: CoreBlox Token Service Policy](#)

[General](#)
[Users](#)
[Rules](#)
[Expression](#)

• = Required



User Directories	
<b>Demo User Directory</b>	
Allow Nested Groups <input type="checkbox"/> AND Users/Groups <input type="checkbox"/>	
<b>Name</b>	<b>User Class</b> <b>Exclude</b>
No results.	
<a href="#">Add Members</a>	<a href="#">Add Entry</a> <a href="#">Add All</a>

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

**Modify Policy: CoreBlox Token Service Policy**[Domains](#) > [Modify Domain: CoreBlox Token Service Domain](#) > [Modify Policy: CoreBlox Token Service Policy](#)**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				

- 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 `coreblox_tokenservices_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.

Parameter	Default	Description
ServiceAgentName	coreblox_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 CoreBlox 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 smreghost.bat (Windows) or smreghost.sh (Linux) run the following command respective of you OS and with the following syntax:
  - a. Windows: smreghost.bat -i ps\_ip\_addr -u sm\_admin -p sm\_admin\_pw -hn trusted\_host\_name -hc sm\_hco
  - b. Linux: ./smregshost.sh -i ps\_ip\_addr -u sm\_admin -p sm\_admin\_pw -hn trusted\_host\_name -hc sm\_hco
  - c. Example (Linux): ./smregshost.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.

## CoreBlox Token Service Configuration Parameters

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

### cts.properties File

The CTS configuration is store in the *cts.properties* file under the *config* directory.

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

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

## Chapter 4: 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 5: Configuring Mutual Client Certificate Authentication

To configure the CoreBlox 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]: coreblox.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=coreblox.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 CoreBlox Token Service is being executed. This certificate will be used as a trusted certificate in the keystore used by the CoreBlox 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 CoreBlox 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]: CoreBlox Token Service
> What is the name of your organizational unit?
    [Unknown]: Services
> What is the name of your organization?
    [Unknown]: coreblox.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=CoreBlox Token Service, OU=Services, O=coreblox.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=coreblox.com, L=New York, ST=NY, C=US
Issuer: CN=Client Certificate, OU=Client Services, O=coreblox.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:E0: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=CoreBlox Token Service, OU=Services, O=coreblox.com, L=New York, ST=NY, C=US
Issuer: CN=CoreBlox Token Service, OU=Services, O=coreblox.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