

# Deploy And Troubleshoot Authorization Code Grant Flow - OAuth Enhancement: Cisco Collaboration Solutions 12.0

## Contents

[Introduction](#)

[Prerequisites](#)

[Requirements](#)

[Components Used](#)

[Background Information](#)

[Feature Highlights](#)

[Important Considerations](#)

[Elements of Authorization Code Grant Flow](#)

[Configure](#)

[Network Diagram](#)

[Refresh Tokens](#)

[Revoke Refresh Tokens](#)

[Verify](#)

[Troubleshoot](#)

[Related Information](#)

## Introduction

This document describes how Authorization Code Grant flow is based on refresh token in order to improve Jabber User Experience across various devices, especially for Jabber on Mobile.

## Prerequisites

### Requirements

Cisco recommends that you have knowledge of these topics:

- Cisco Unified Communications Manager (CUCM) 12.0 version
- Single Sign On (SSO)/SAML
- Cisco Jabber
- Microsoft ADFS
- Identity Provider (IdP)

In order to get more information on these topics, refer to these links:

- [SAML SSO Deployment Guide for Cisco Unified Communications](#)
- [Unified Communications Manager SAML SSO Configuration Example:](#)
- [AD FS Version 2.0 Setup for SAML SSO Configuration Example:](#)

## Components Used

The information in this document is based on these software:

- Microsoft ADFS ( IdP)
- LDAP Active Directory
- Cisco Jabber Client
- CUCM 12.0

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

## Background Information

As of today, Jabber SSO flow with Infrastructure is based on Implicit Grant Flow where CUCM Authz service allocates the short-lived access tokens.

Post access token expiry, CUCM redirects Jabber to IdP for re-authentication.

This leads to a bad user experience, especially with jabber on mobile where the user is asked to enter credentials frequently.

Security Re-architecture Solution also proposes Authorization Code Grant Flow (with the use of Refresh Tokens approach (extendible to End Points/Other Collaboration Apps)) for the unification of Jabber and End Point log in flow for both SSO and Non-SSO scenarios.

## Feature Highlights

- Authorization Code Grant flow is based on refresh token (extendible to End Points/Other Collaboration Apps) in order to improve Jabber User Experience across various devices, especially for Jabber on Mobile.
- Supports Self Contained Signed and Encrypted OAuth Tokens to allow various collaboration applications to validate and respond to client resource requests.
- The implicit grant flow model is retained which allows backwards compatibility. This also allows a seamless path for other clients (like RTMT) who have not moved to Authorization Code Grant flow.

## Important Considerations

- Implementation such that the old jabber client can work with the new CUCM (since it supports both implicit grant and authorization code grant flows). Also, the new jabber can work with the old CUCM. Jabber can determine if CUCM supports Authorization Code Grant flow and only if it supports this model, it switches and uses implicit grant flow.
- The AuthZ service runs on the CUCM server.
- AuthZ supports only Implicit Grant Flow. This means there was no refresh token/offline access token. Each time client wanted a new Access token, the user needs to re-authenticate with the IdP.
- Access Tokens were issued only if your deployment is SSO enabled. Non-SSO deployments

- didn't work in this case and access Tokens were not used on all interfaces consistently.
- Access Tokens are not self-contained but rather retained in the memory of the server that issued them. If CUCM1 issued the access token, it can be verified only by CUCM1. If the client tries to access service on CUCM2, CUCM2 needs to validate that token on CUCM1. Network delays (proxy mode).
  - User experience on mobile clients is very bad since the user has to re-enter credentials on an alpha-numeric keypad when the user re-authenticates with the IdP (typically running from 1 hour to 8 hours which depends on several factors).
  - Clients that speak to multiple applications over multiple interfaces need to maintain multiple credentials/blocks. No seamless support for same user log in from 2 similar clients. For instance, user A logs in from jabber instances that run on 2 different iPhones.
  - AuthZ to support both SSO and Non-SSO deployments.
  - AuthZ to support implicit grant flow + authorization code grant flow. As it is **backward compatible**, it allows clients like **RTMT** to continue work until they adapt.
  - With Authorization code grant flow, AuthZ issues access token and refresh token. The refresh token can be used to get another access token without the need for authentication.
  - Access Tokens are self-contained, signed and encrypted and use the JWT (JSON web tokens) standard (RFC compliant).
  - Signing and encryption keys are common to the cluster. Any server in the cluster can verify the access token. There is no need to maintain in memory.
  - the service that runs on CUCM 12.0 is the centralized Authentication Server in the cluster.
  - Refresh tokens are stored in Database (DB). Admin needs to be able to revoke it, if required. Revocation is based on userid or userid & clientID.
  - Signed access tokens allow different products to validate access tokens without the need to store them. Configurable access token and refresh token lifetimes (default 1 hour and 60 days respectively).
  - JWT format is aligned with Spark which allows synergies in the future with Spark Hybrid services.
  - Support for the same user logs in from 2 similar devices. Eg: User A can log in from jabber instances that runs on 2 different iPhones.

## Elements of Authorization Code Grant Flow

- Auth Z Server
- Encryption Keys
- Signing Keys
- Refresh Tokens

## Configure

This feature is not enabled by default.

Step 1. In order to enable this feature, navigate to **System > Enterprise Parameters**.

Step 2. Set the param **OAuth with Refresh Login Flow** to Enabled as shown in the image.

SSO and OAuth Configuration		
OAuth Access Token Expiry Timer (minutes) *	60	60
OAuth Refresh Token Expiry Timer (days) *	60	60
Redirect URIs for Third Party SSO Client		
SSO Login Behavior for iOS *	Use embedded browser (WebView)	Use embedded browser (WebView)
OAuth with Refresh Login Flow *	Enabled	Disabled
Use SSO for RTMT *	True	True

- Access token is signed and encrypted. Signing and encryption key is common to the cluster. This means any node in the cluster can validate the access token.
- The access token is in JWT format (RFC 7519).
- Access tokens re-use enterprise parameter (OAuth Access Token Expiry timer), which is applicable for both old token and new token formats.
- Default value - 60 Mins.
- Minimum Value - 1 Min.
- Maximum Value - 1440 Mins

```
eyJhbGciOiJIUzU1NiIsInR5cCI6IkpXVCIsImtpZCI6IjkhMGQ1MzI0LWY0ZjAtNGIwYi04MTF1LTRhNTlmZGI2YjcyMjppj
Mjc3MGM5N2JkYTlkMzRmZDA1YTdlYTFhZWQzZTU0Y2E4MGJkZDdlZTM1ZDk3MDNiNjBiNTQ5MTBiZDQ0ODRiIn0.eyJwcm12
YXRlIjoizXlKaGJHY2lPaUprYVhJaUxDSmpkSGtpT2lKSlYxUWlMQ0psYm1NaU9pSkJNVEk0UTBKRExVaFRNaUySWl3aWEy
bGtJam9pT0dRd1pEVXpNalF0WmpSbU1DMDBZakJpTFRneE1XVXROR0UxT1daallqWmlOek15T2lVd1ptUm1ZMk16WlRRMU5E
RTFOV0ZpTkrJek5tRTJOMlV4T0RChU1qWmxZMk13WXpJee56SXLOREJtWlRFellXWXlOak14TkRkalpHVXpNR1l3TjJJaWZR
Li5xQWd6aGdRaTVMmKdlaDl5V2RvN25nLmdMTHNpaTRjQk50c1NEUXRJTE51RWRnWTl4WkVJvcJ4YzBaeTFGQjZQNmNzWWJf
ZkRnaDRZby04V1NaNjUzdXowbnFOalpXT1E1dGdnYW9qMlp6ZFk2ZzN2SWFhbF9JWUtNdKNIWNNscmt4YUFGTk5MWEXLQlJm
aTA2LVk2V3l1dUdxNmpNwK5DbnlKXlpTbUpkVFQwc1Z4RTdGTXVxaUJSMElrRGdyVDDvOFNXMEY5cXFadndEZDJSaDdqNkRJ
WGdks3VtOwltU2xNU1pjejhueVdic01Udk5yMWY0M25VenJzMHk5WwN6NnBDX0czZmlWYjJsX2VWLVFkcFh4TUo2bnZodXcy
djRiUGVkm3VMQlpaVWloQ3B6TUVDdW5NMlh1TVBrTGdlS1NqWG44aGhPRFNVcWlWQ0Uta3RzdRbc2Q0RnJxcGNxWlZiS0Zi
VTFRbU0wV2pMYVJtUk9IV1lQVkc0a3FBdTRWalVMUzVCRWszNnZ4Nmp3U3BMUy1IdTcwbVRNcmR3dmV5Q2ZOYkhyT0FlVmVv
ekFIR3JqdGlmaFpmSFVUTWZiNkMtX2tOQVJGQWdDclZTZY0wUzlxblJvTWVkcUENETEE4MDJiaWwtNDJjOC15Mwo4X1FVaC02
UUtCV2dodVd4VWtBODRpekFFaWl0QTlsSHFKM3Nxd2JFNURkZmhIay05bTJfTTN5MWlWVkdORVQ3ZW9XVDBqWllnRGRBqjFz
UGwxLTlaSFNYmsydTE3SkJVRV9FOXl0V0tWMNBqWgtin0LQSWgtQ3JWQTZkcVdQRHVIbmx1V19wblNlYnYtTkZVbGQ0WEY3
cmZLYmQySlg4eUhhX05pOVVVUnUwZVdsNWxGRUVabklubmFKZEdHLUZrb3VuN2xHSFlwSE4ydXVudmRnOHZVZzZsa0JPbmoz
eUFjclZTMGxKc1NwdUxYFYldwd2c4YjdBdDM3d3AtMWT2Y1ZQaWpCQ1lCV181d2JzbTFYd2k4MVc2WHVpNzZmZmV3cEJVVQnBf
T2VRNzQ2ZXJJKekNUUFZCYUpZUGJuzWEtdFhsU3RmZzBGeVRmbnhnX1Vzazl3QXJkemE4c204T0FQaWmXZmFQOG0uUTdFN0FV
X2xUVnNmZFI2bnkydUdhQSJ9.u2fJrVA55NQC3esPb4kcodt5rnjcl0-5uEDdUf-
KnCYEPBZ7t2CTsMMVVE3nfRhM39MfTlNS-qVOVpuoW_51NyAENXQMxfxlU9aXp944QiU1OeFQKj_g-
n2dEINRStbtUc3KMKqtz38Bff1g2Z51sdlnBn4XyVWPgGCF4XSfsFIa9fF051awQ0LcCv6YQTGer_6nk7t6F1MzPzBzZjala
bpm--6LNSzjPftEiexpD2oXvW8V10Z9ggNk5Pn3Ne4RzqK09J9WChaJSXkTTE5G39EZcePmVNTcbayq-
L2pAK5weDa2k4uYmfAQAwcToHUrWk3yilwqjHAamcG-CoipZQ
```

OAuth Refresh Token Expiry Timer" parameter in enterprise parameters page in CUCM.  
 Path: System -> Enterprise parameters  
 Values are integers ranging from 1 - 90  
 Minimum lifetime = 1 Day  
 Default lifetime = 60 days  
 Maximum lifetime = 90 days

New access token is issued each time client requests for one. The old one continues to be valid as long as:

- Signing/Encryption keys have not changed
- Validity (stored inside the token) breaks.
- JSON web-tokens: consist of three parts, separated by dots, which are: Header, Payload & Signature.

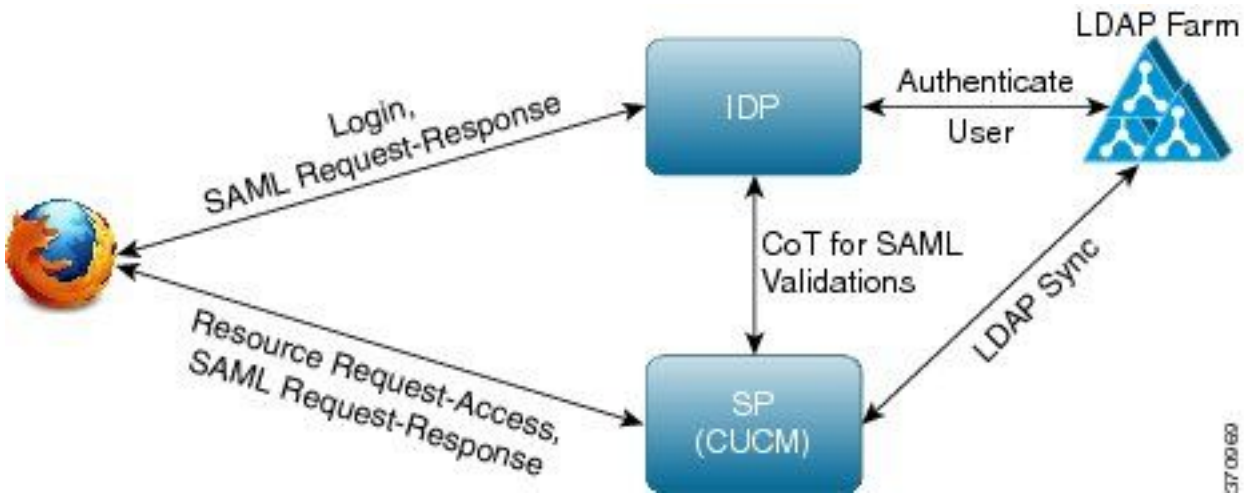
Sample access token:

- At the start of the token that is highlighted in bold is the Header.

- Middle part is the Payload.
- At the end, if the token is highlighted in bold then it is the Signature.

## Network Diagram

Here is a high level overview of the call flow involved:



## Refresh Tokens

- Refresh token are signed.
- Refresh token is stored in **refreshtokendetails** table in the database as a hash value of itself. This is to prevent replication by DB as it can be picked by someone. To review the table you can run:

```
run sql select * from refreshtokendetails
```

Or with a readable validity date:

```
run sql select pkid,refreshtokenindex,userid,clientid,dbinfo('utc_to_datetime',validity) as validity,state from refreshtokendetails
```

```
admin:run sql select * from refreshtokendetails
```

pkid	refreshtokenindex	userid	clientid	validity	state
173e2283-1...	65483476618891...	bvanturn	Clb4b...	2019-01-05 14:11:46	1080686546
cd2c634c-7...	0bf6b2989db114...	bvanturn	Clb4b...	2019-01-05 14:28:41	569144456
a3706858-b...	b4800f20dbfe0e...	bvanturn	Clb4b...	2019-01-05 14:38:12	1146722445

**Warning:** Refresh token is flushed from DB when the validity is expired. Timer thread runs at 2 am everyday (not configurable via UI, but can be modified via remote support account). If the table has a large number of access tokens, that are invalid and need to be flushed out. This can cause a CPU spike.

Sample refresh token:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjkhMGQ1MzI0LWY0ZjAtNGIwYi04MTFlLTRhNTlmZGI2YjcyMjppMjc3MGM5N2JkYTlkMzRmZDA1YTdlYTFhZWQzZTU0Y2E4MGJkZDdlZTM1ZDk3MDNiNjBiNTQ5MTBiZDQ0ODRiIn0.eyJleHAiOiJlMDI2MjAwNTIzImlzcyI6IjkhMGQ1MzI0LWY0ZjAtNGIwYi04MTFlLTRhNTlmZGI2YjcyMjppMjIwYzI0MTY4MzUyZjZjMzdkMzZmNDM4ZWYwZWYyN2MwOTM4YWRjNjIyYmUwYzAzZDE2OWYyYSJ9.creRusfwSYAMAtttS2FIPAgIVvCiREvznzlouxeYGVndalJlMa-ZpRqv8FOBrSywqEyulr1-Tem8XGGQCuvFaqO9IkhJqSYz3zvFvvySWzDhl_pPyWlIQteAhLlGaQkue6a5ZegeHRp1sjEcZKMLC6H68CHCfletn5-
```

## Revoke Refresh Tokens

Admin has the capability to revoke all refresh tokens for a user or device-only refresh tokens for a user through **userID** or **userID** and **ClientID**.

In order to revoke device-based RTs for a user:

- revoke RT for user xyz and device identified by client\_id abc.
- [https://cucm-193:8443/ssosp/token/revoke?user\\_id=xyz&client\\_id=abc](https://cucm-193:8443/ssosp/token/revoke?user_id=xyz&client_id=abc)

Signing and Encryption keys




- Signing key is RSA based, who has public/private key pair.
- The encryption key is a symmetric key.
- These keys are created only on the publisher and are distributed across all the nodes in the cluster.
- Both the signing key and encryption key can be re-generated, with the use of the options listed. However, this must be done only if the administrator believes that the keys have been compromised. The impact of the re-generation of either of these keys is that all the access tokens issued by AuthZ service becomes invalid.
- Signing keys can be re-generated with UI and CLI.
- Encryption keys can be regenerated only with CLI.

The regeneration of Authz certs (signing key) from the **Cisco Unified OS Administration** page on CUCM is as shown in the image.

Certificate Details(Self-signed) - Internet Explorer provided by Cisco Systems, Inc.


https://10.77.29.184/cmplatform/certificateEdit.do?cert=/usr/local/platform/.security/authz/certs/authz.j ✖ Certificate error

### Certificate Details for AUTHZ\_CUCM-184, authz

 Regenerate
  Download .PEM File
  Download .DER File

---

**Status**

 Status: Ready

---

**Certificate Settings**

File Name	authz.pem
Certificate Purpose	authz
Certificate Type	certs
Certificate Group	product-cpi
Description(friendly name)	Self-signed certificate generated by system

---

**Certificate File Data**

```

[
[
Version: V3
Subject: L=i, ST=i, CN=AUTHZ_CUCM-184, OU=i, O=i, C=IN
Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11

Key: CiscoJ RSA Public Key, 2048 bits
modulus:
310088952412132774650041525392629167237879710935753621934671843
216346326898490353644164813514840735197164588955185219996734516
256663568507413849247845292675452179850077675141884383314726763
520023902784651553941826511494962731151521090167892375623419501
739811988911210916820812069748957615302991414362015465824669063
319779866264424936428249029193098223306846888723560182717860238
318402233050626785154245146789308145325775236137097363983609689
  
```

The regeneration of the Authz signing key with the use of the CLI command is as shown in the image.

```
CUCM-184 login: admin
Password:
Last login: Tue Nov 15 15:43:52 on tty1
Command Line Interface is starting up, please wait ...
```

```
Welcome to the Platform Command Line Interface
```

```
VMware Installation:
 1 vCPU: Intel(R) Xeon(R) CPU E5-2643 0 @ 3.30GHz
Disk 1: 80GB, Partitions aligned
6144 Mbytes RAM
```

```
admin:set ke
admin:set key regen authz signing
```

```
WARNING: This operation will regenerate the Authorization Service signing key and restart the Authorization Service on all the nodes. It is recommended that this command be run off-hours to avoid end user impact.
```

```
Proceed with regeneration (yes/no)? yes
```

```
signing key for the Authorization service generated successfully.
```

```
admin:_
```

Admin can display authz signing and encryption keys with the use of CLI. Hash of the key is displayed rather than original key.

Command to display keys are:

Signing Key: **show key authz signing** and as shown in the image.

```
admin:show key authz signing
authz signing key with checksum: a155d81be734850226f990a62816f1ae last synced on: 06/09/2017 13:04:47
```

Encryption Key: **show key authz encryption** and as shown in the image.

```
admin:show key authz encryption
authz encryption key with checksum: 88edce92173e33f9cedbbfb09cd0e8c4 last synced on: 06/14/2017 16:22:06
```

**Note:** The signing authz and encryption authz are always different.

## Verify

Use this section in order to confirm that your configuration works properly.

When it is intended to use OAuth on the Cisco Unity Connection (CUC) server, the network administrator must perform two steps.

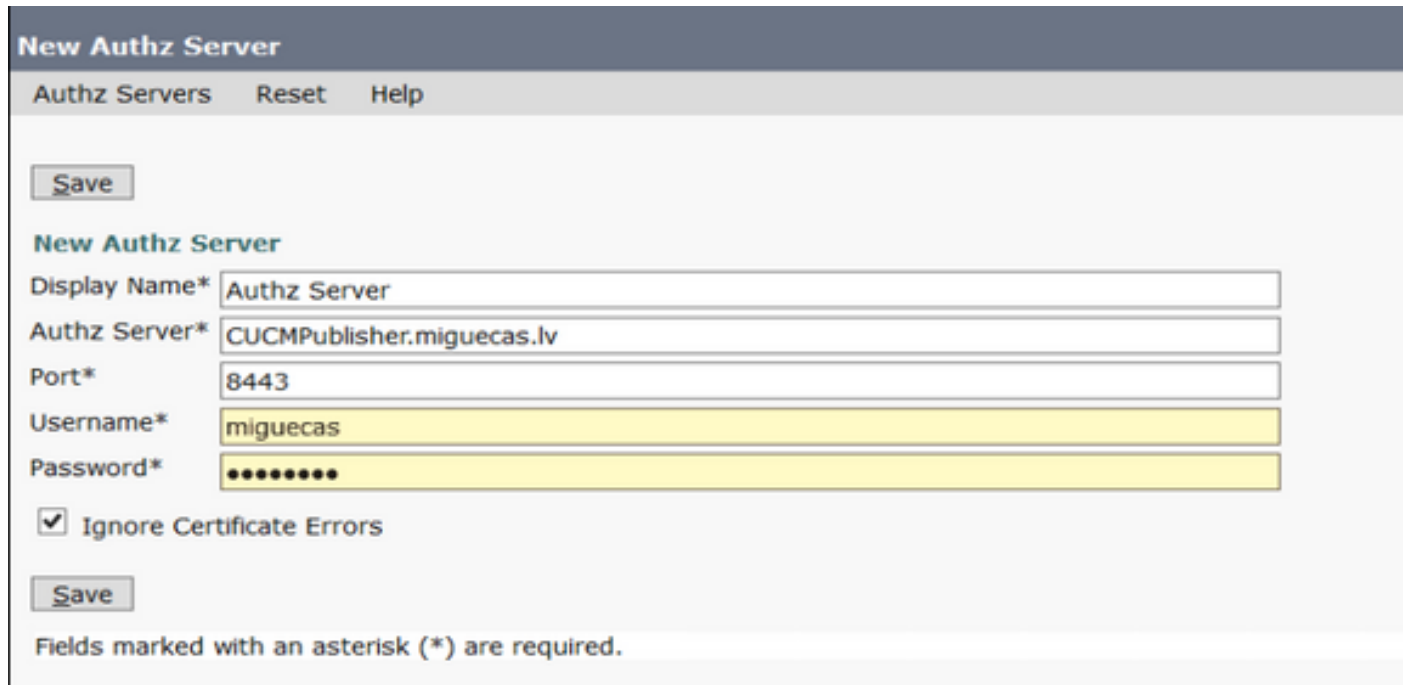
Step 1. Configure the Unity Connection Server to fetch the OAuth Token signing and encryption keys from the CUCM.

Step 2. Enable OAuth Services on the CUC Server.



**Note:** To fetch the signing and encryption keys, Unity must be configured with the CUCM host details and a user account enabled of the CUCM AXL Access. If this is not configured, the Unity Server cannot retrieve the OAuth Token from the CUCM and the voicemail log in for the users cannot be available.

Navigate to **Cisco Unity Connection Administration > System Settings > Authz Servers**



**New Authz Server**

Authz Servers   Reset   Help

**New Authz Server**

Display Name\*

Authz Server\*

Port\*

Username\*

Password\*

Ignore Certificate Errors

Fields marked with an asterisk (\*) are required.

## Troubleshoot

This section provides the information you can use in order to troubleshoot your configuration.

**Note:** If OAuth is used and the Cisco Jabber users are unable to log in, always review the signing and encryption keys from the CUCM and Instant Messaging and Presence (IM&P) Servers.

The network administrators need to **run** these two commands on all the CUCM and IM&P nodes:

- **show key authz signing**
- **show key authz encryption**

If the signing authz and encryption authz outputs do not match across all the nodes, they require to be regenerated. In order to perform that, these two commands need to be run on all the CUCM and IM&P nodes:

- **set key regen authz encryption**
- **set key regen authz signing**

Afterwards, the **Cisco Tomcat** service needs to be restarted on all the nodes.

Along with the keys' mismatch, this error line can be found on the Cisco Jabber logs:

```
2021-03-30 14:21:49,631 WARN [0x0000264c] [vices\impl\system\SingleSignOn.cpp(1186)] [Single-Sign-On-
```

Logger] [CSFUnified::SingleSignOn::Impl::handleRefreshTokenFailure] - **Failed to get valid access token from refresh token, maybe server issue.**

The sso app logs are generated in these locations:

- **file view activelog platform/log/ssoApp.log** This does not require any trace configuration for log collection. Every time SSO App operation is done, a new log entries are generated in ssoApp.log file.
- **SSOSP logs: file list activelog tomcat/logs/ssosp/log4j**  
Every time sso is enabled, a new log file is created at this location with name, **ssosp00XXX.log**. Any other SSO operation and all Oauth operations are also logged into this file.
- **Certificate logs: file list activelog platform/log/certMgmt\*.log**  
Everytime AuthZ certificate is regenerated (UI or CLI), a new log file is generated for this event.  
For authz encryption key re-generation, a new log file is generated for this event.

## Related Information

[Deploying OAuth with Cisco Collaboration Solution Release 12.0](#)