Déployer Et Dépanner Le Flux De Subvention De Code D'Autorisation - Amélioration OAuth : Solutions de collaboration Cisco 12.0

Contenu

Introduction

Ce document décrit comment le flux d'octroi de code d'autorisation est basé sur un jeton d'actualisation afin d'améliorer l'expérience utilisateur Jabber sur différents périphériques, en particulier pour Jabber sur Mobile.

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- Cisco Unified Communications Manager (CUCM) version 12.0
- Authentification unique (SSO)/SAML
- Cisco Jabber
- Microsoft ADFS
- Fournisseur d'identité (IdP)

Pour obtenir plus d'informations sur ces sujets, reportez-vous aux liens suivants :

- Guide de déploiement SAML SSO pour Cisco Unified Communications
- Exemple de configuration d'Unified Communications Manager SAML SSO :

- Exemple de configuration de l'installation d'AD FS version 2.0 pour SAML SSO :

Components Used

Les informations de ce document sont basées sur ce logiciel :

- Microsoft ADFS (IdP)
- Annuaire Active Directory LDAP
- Client Cisco Jabber
- 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. Si votre réseau est en ligne, assurez-vous de bien comprendre l'incidence possible des commandes.

Informations générales

Àpartir d'aujourd'hui, le flux SSO Jabber avec infrastructure est basé sur le flux de subvention implicite où le service CUCM Authz alloue les jetons d'accès de courte durée.

Après expiration du jeton d'accès, CUCM redirige Jabber vers IdP pour la réauthentification.

Cela entraîne une mauvaise expérience utilisateur, en particulier avec jabber sur mobile où l'utilisateur est fréquemment invité à saisir des informations d'identification.

Security Re-architecture Solution propose également un flux d'autorisation de code (avec l'utilisation de l'approche Refresh Tokens (extensible aux terminaux/autres applications de collaboration)) pour l'unification du flux de connexion Jabber et End Point pour les scénarios SSO et non SSO.

Caractéristiques principales

- Le flux d'octroi de code d'autorisation est basé sur un jeton d'actualisation (extensible aux terminaux/autres applications de collaboration) afin d'améliorer l'expérience utilisateur de Jabber sur différents périphériques, en particulier pour Jabber sur mobile.
- Prend en charge les jetons OAuth signés et cryptés autonomes pour permettre à diverses applications de collaboration de valider et de répondre aux demandes de ressources client.
- Le modèle de flux de subvention implicite est conservé, ce qui permet une compatibilité descendante. Cela permet également un chemin d'accès transparent pour les autres clients (comme RTMT) qui n'ont pas été déplacés vers le flux d'octroi de code d'autorisation.

Considérations importantes

- Mise en oeuvre telle que l'ancien client jabber puisse travailler avec le nouveau CUCM (puisqu'il prend en charge les flux implicites de subvention et de code d'autorisation). De plus, le nouveau jabber peut fonctionner avec l'ancien CUCM. Jabber peut déterminer si CUCM prend en charge le flux d'octroi de code d'autorisation et uniquement s'il prend en charge ce modèle, il bascule et utilise le flux d'octroi implicite.
- Le service AuthZ s'exécute sur le serveur CUCM.

- AuthZ prend uniquement en charge le flux de subvention implicite. Cela signifie qu'il n'y a pas eu de jeton d'actualisation/jeton d'accès hors connexion. Chaque fois que le client souhaite un nouveau jeton d'accès, l'utilisateur doit se réauthentifier avec l'IDP.
- Les jetons d'accès ont été émis uniquement si votre déploiement est activé par SSO. Les déploiements non SSO n'ont pas fonctionné dans ce cas et les jetons d'accès n'ont pas été utilisés de manière cohérente sur toutes les interfaces.
- Les jetons d'accès ne sont pas autonomes mais sont plutôt conservés dans la mémoire du serveur qui les a émis. Si CUCM1 a émis le jeton d'accès, il ne peut être vérifié que par CUCM1. Si le client tente d'accéder au service sur CUCM2, CUCM2 doit valider ce jeton sur CUCM1. Délais réseau (mode proxy).
- L'expérience utilisateur sur les clients mobiles est très mauvaise, car l'utilisateur doit saisir à nouveau des informations d'identification sur un clavier alphanumérique lorsque l'utilisateur se réauthentifie avec l'IDP (généralement exécuté de 1 heure à 8 heures, selon plusieurs facteurs).
- Les clients qui parlent à plusieurs applications sur plusieurs interfaces doivent conserver plusieurs informations d'identification/blocs. Aucune prise en charge transparente pour le même utilisateur se connecte à partir de 2 clients similaires. Par exemple, l'utilisateur A se connecte à partir d'instances jabber qui s'exécutent sur 2 iPhone différents.
- AuthZ pour prendre en charge les déploiements SSO et non SSO.
- AuthZ pour prendre en charge le flux de subvention implicite + le flux de subvention du code d'autorisation. Comme il est **rétrocompatible,** il permet à des clients comme **RTMT** de continuer à travailler jusqu'à ce qu'ils s'adaptent.
- Avec le flux d'octroi de code d'autorisation, AuthZ émet le jeton d'accès et le jeton d'actualisation. Le jeton d'actualisation peut être utilisé pour obtenir un autre jeton d'accès sans avoir besoin d'authentification.
- Les jetons d'accès sont autonomes, signés et chiffrés et utilisent la norme JWT (JSON Web Tokens) (conforme RFC).
- Les clés de signature et de chiffrement sont communes au cluster. Tout serveur du cluster peut vérifier le jeton d'accès. Il n'est pas nécessaire de garder en mémoire.
- le service qui s'exécute sur CUCM 12.0 est le serveur d'authentification centralisé dans le cluster.
- Les jetons d'actualisation sont stockés dans la base de données (DB). L'administrateur doit être en mesure de le révoquer, si nécessaire. La révocation est basée sur userid ou userid & clientID.
- Les jetons d'accès signés permettent à différents produits de valider les jetons d'accès sans devoir les stocker. Durée de vie du jeton d'accès configurable et du jeton d'actualisation (par défaut, 1 heure et 60 jours respectivement).
- Le format JWT est aligné sur Spark, ce qui permet des synergies futures avec les services hybrides Spark.
- La prise en charge du même utilisateur se connecte à partir de 2 périphériques similaires. Par exemple : L'utilisateur A peut se connecter à partir d'instances jabber qui s'exécutent sur 2 iPhone différents.

Éléments du flux de subvention du code d'autorisation

- Serveur Auth Z
- Clés de chiffrement

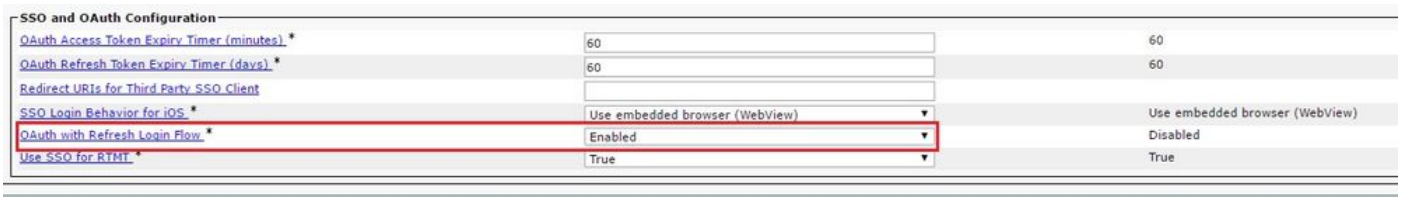
- Clés de signature
- Actualiser les jetons

Configuration

Cette fonctionnalité n'est pas activée par défaut.

Étape 1. Afin d'activer cette fonctionnalité, accédez à **System > Enterprise Parameters**.

Étape 2. Définissez le paramètre **OAuth avec le flux de connexion** de **rafraîchissement** sur Activé, comme indiqué dans l'image.



- Le jeton d'accès est signé et chiffré. La clé de signature et de chiffrement est commune au cluster. Cela signifie que tout noeud du cluster peut valider le jeton d'accès.
- Le jeton d'accès est au format JWT (RFC 7519).
- Les jetons d'accès réutilisent le paramètre d'entreprise (minuteur d'expiration du jeton d'accès OAuth), qui est applicable aux formats de jeton ancien et nouveau.
- Valeur par défaut : 60 min.
- Valeur minimale : 1 min.
- Valeur maximale : 1 440 min


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjhkMGQ1MzI0LWY0ZjAtNGIwYi04MTFlLTRhNTlmZGI2YjcyMjpj
Mjc3MGM5N2JkYTlkMzRmZDA1YTdlYTFhZWQzZTU0Y2E4MGJkZDdlZTM1ZDk3MDNiNjBiNTQ5MTBiZDQ0ODRiIn0.eyJwcml2
YXRlIjoiZXlKaGJHY2lPaUprYVhJaUxDSmpkSGtpT2lKS1YxUWlMQ0psYm1NaU9pSkJNVEk0UTBKRExVaFRNalUySWl3aWEy
bGtJam9pT0dRd1pEVXpNalF0WmpSbU1DMDBZakJpTFRneE1XVXROR0UxT1daa1lqWmlOekl5T21Vd1ptUm1ZMk16WlRRMU5E
RTFOV0ZpTkRJek5tRTJOMlV4T0RCbU1qWmxZMkl3WXpJeE56SXlOREJtWlRFellXWXlOak14TkRkalpHVXpNR1l3TjJJaWZR
Li5xQWd6aGdRaTVMMkdlaDl5V2RvN25nLmdMTHNpaTRjQk50c1NEUXRJTE51RWRnWTl4WkJVczJ4YzBaeTFGQjZQNmNzWWJf
ZkRnaDRZby04V1NaNjUzdXowbnFOalpXT1E1dGdnYW9qMlp6ZFk2ZzN2SWFHbF9JWUtNdkNIWWNscmt4YUFGTk5MWExLQlJm
aTA2LVk2V3l1dUdxNmpNWk5DbnlKX1pTbUpkVFQwc1Z4RTdGTXVxaUJsMElrRGdyVDdvOFNXMEY5cXFadndEZDJSaDdqNkRJ
WGdkS3VtOWltU2xNU1pjejhueVdic01Udk5yMWY0M25VenJzMHk5WWN6NnBDX0czZmlWYjJsX2VWLVFkcFh4TUo2bnZodXcy
djRiUGVkM3VMQlpaVW1oQ3B6TUVDdW5NMlh1TVBrTGdlS1NqWG44aGhPRFNVcW1WQ0Uta3RZdnRBc2Q0RnJxcGNxWlZiS0Zi
VTFRbU0wV2pMYVJtUk9IVllQVkc0a3FBdTRWalVMUzVCRWszNnZ4Nmp3U3BMUy1IdTcwbVRNcmR3dmV5Q2ZOYkhyT0FlVmVv
ekFIR3JqdGlmaFpmSFVUTWZiNkMtX2tOQVJGQWdDclZTZy0wUzlxb1JvTWVkUENETEE4MDJiaWwtNDJjOC15MWo4X1FVaC02
UUtCV2dodVd4VWtBODRpekFFaWl0QTlsSHFKM3Nxd2JFNURkZmhIay05bTJfTTN5MWlWVkdoRVQ3ZW9XVDBqWllnRGRBQjFz
UGwxLTlaSFNYYmsydTE3SkJVRV9FOXI0V0tWMnBqWGtiN0lQSWgtQ3JWQTZkcVdQRHVIbmx1V19wblNLYnYtTkZVbGQ0WEY3
cmZLYmQySlg4eUhhX05pOVVVUnUwZVdsNWxGRUVabklubmFKZEdHLUZrb3VuN2xHSFlwSE4ydXVudmRnOHZVZzZsa0JPbmoz
eUFjc1ZTMGxKc1NWdUxFYldwd2c4YjdBdDM3d3AtMWt2Y1ZQaWpCQ1lCV181d2JzbTFYd2k4MVc2WHVpNzMzQVg3cEJVQnBf
T2VRNzQ2ZXJJekNUUFZCYUpZUGJuZWEtdFhsU3RmZzBGeVRmbnhnX1Vzazl3QXJkemE4c204T0FQaWMxZmFQOG0uUTdFN0FV
X2xUVnNmZFI2bnkydUdhQSJ9.u2fJrVA55NQC3esPb4kcodt5rnjc1o-5uEDdUf-
KnCYEPBZ7t2CTsMMVVE3nfRhM39MfTlNS-qVOVpuoW_51NYaENXQMxfxlU9aXp944QiU1OeFQKj_g-
n2dEINRStbtUc3KMKqtz38BFf1g2Z51sdlnBn4XyVWPgGCf4XSfsFIa9fF051awQ0LcCv6YQTGer_6nk7t6F1MzPzBZzja1a
bpm--6LNSzjPftEiexpD2oXvW8Vl0Z9ggNk5Pn3Ne4RzqK09J9WChaJSXkTTE5G39EZcePmVNtcbayq-
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
```

Un nouveau jeton d'accès est émis chaque fois que le client en fait la demande. L'ancienne demeure valable aussi longtemps que :

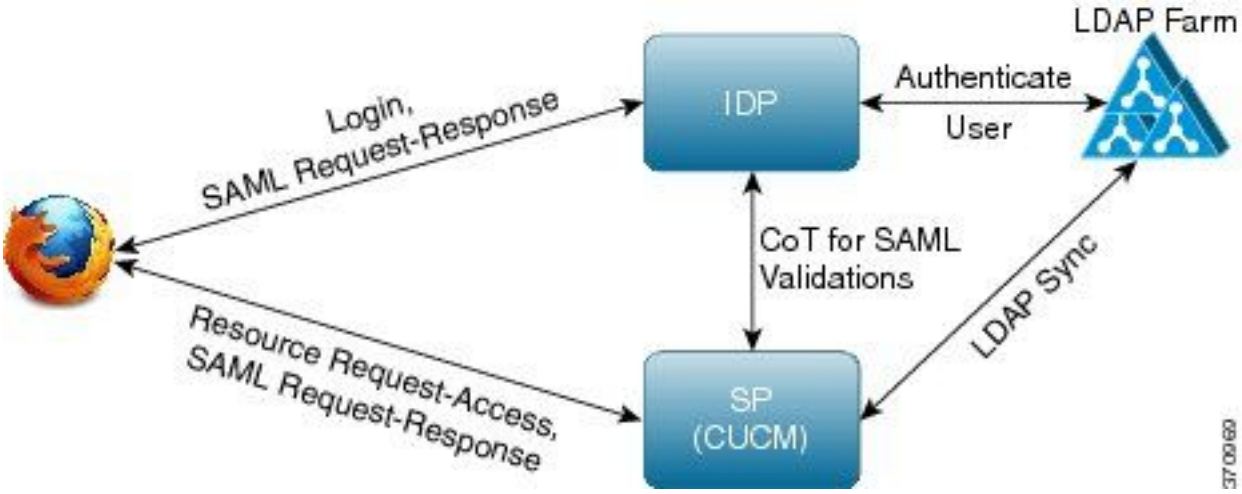
- Les clés de signature/chiffrement n'ont pas été modifiées
- La validité (stockée dans le jeton) se brise.
- JSON Web-Tokens : se composent de trois parties, séparées par des points, qui sont : En-tête, charge utile et signature.

Exemple de jeton d'accès :

- Au début du jeton mis en surbrillance en gras se trouve l'en-tête.
- La partie centrale est la charge utile.
- Àla fin, si le jeton est mis en surbrillance en gras, il s'agit de la signature.

Diagramme du réseau

Voici une présentation générale du flux d'appels concerné :



Actualiser les jetons

- Le jeton d'actualisation est signé.
- Le jeton d'actualisation est stocké dans la table **de détails d'actualisation** de la base de données en tant que valeur de hachage elle-même. Il s'agit d'empêcher la réplication par DB car elle peut être choisie par quelqu'un. Pour consulter la table, vous pouvez exécuter :
  `run sql select * from refreshtokendetails`
  Ou avec une date de validité lisible :

```
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 C1b4b... 2019-01-05 14:11:46 1080686546
cd2c634c-7... 0bf6b2989db114... bvanturn C1b4b... 2019-01-05 14:28:41 569144456
a3706858-b... b4800f20dbfe0e... bvanturn C1b4b... 2019-01-05 14:38:12 1146722445
```

Avertissement : Le jeton d'actualisation est vidé de la base de données lorsque la validité est expirée. Le thread du minuteur s'exécute tous les jours à 2 heures du matin (il n'est pas configurable via l'interface utilisateur, mais peut être modifié via un compte de support distant). Si la table comporte un grand nombre de jetons d'accès, ceux-ci ne sont pas valides et doivent être supprimés. Cela peut provoquer une pointe de CPU.

```
Sample refresh token:

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjhkMGQ1MzI0LWY0ZjAtNGIwYi04MTFlLTRhNTlmZGI2YjcyMjpj
Mjc3MGM5N2JkYTlkMzRmZDA1YTdlYTFhZWQzZTU0Y2E4MGJkZDdlZTM1ZDk3MDNiNjBiNTQ5MTBiZDQ0ODRiIn0.eyJleHAi
OjE1MDI2MjAwNTIsImlzcyI6IjhkMGQ1MzI0LWY0ZjAtNGIwYi04MTFlLTRhNTlmZGI2YjcyMiIsInR5cCI6InVzZXIiLCJ0
aWQiOiJiOTkxMjIxZi1mNDJlLTRlNTItODg3MS1jODc2ZTYzNWRkNWIiLCJjdHlwIjoicmVmcmVzaCIsImNjaWQiOiJDM2Iw
YWZmZWZlMTQzOTA0MTY4M2U5YzJjMzdkMzZmNDM4ZWYwZWYyN2MwOTM4YWRjNjIyNmUwYzAzZDE2OWYyYSJ9.creRusfwSYA
MAtttS2FIPAgIVvCiREvnzlouxeyGVndalJlMa-ZpRqv8FOBrsYwqEyulrl-
TeM8XGGQCUvFaqO9IkhJqSYz3zvFvvySWzDhl_pPyWIQteAhL1GaQkue6a5ZegeHRp1sjEczKMLC6H68CHCfletn5-
j2FNrAUOX99Vg5h4mHvlhfjJEel3dU_rciAIni12e3LOKajkzFxF6W0cXzzujyi2yPbY9gZsp9HoBbkkfThaZQbSlCEpvB3t
7yRfEMIEaHhEUU4M3-uSybuvitUWJnUIdTONiWGRh_fOFR9LV3Iv9J54dbsecpsncc369pYhu5IHwvsglNKEQ
```

Révoquer les jetons d'actualisation

Admin peut révoquer tous les jetons d'actualisation pour un utilisateur ou un périphérique uniquement par l'intermédiaire de **userID** ou **userID** et **ClientID**.

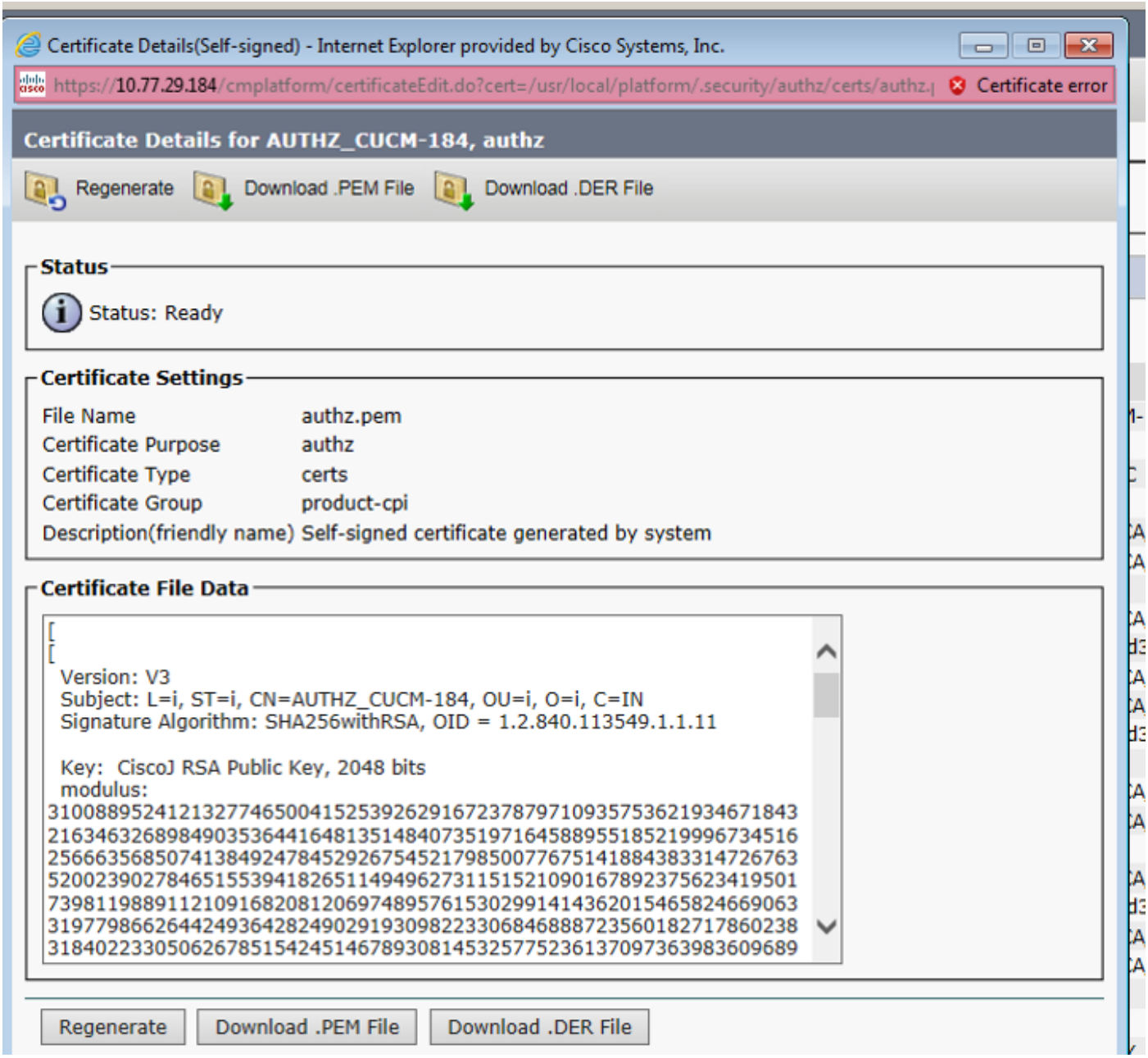
Afin de révoquer les RT basées sur les périphériques pour un utilisateur :

- révoquer RT pour l'utilisateur xyz et le périphérique identifié par client_id abc.
- https://cucm-193:8443/ssosp/token/revoke?user_id=xyz&client_id=abc

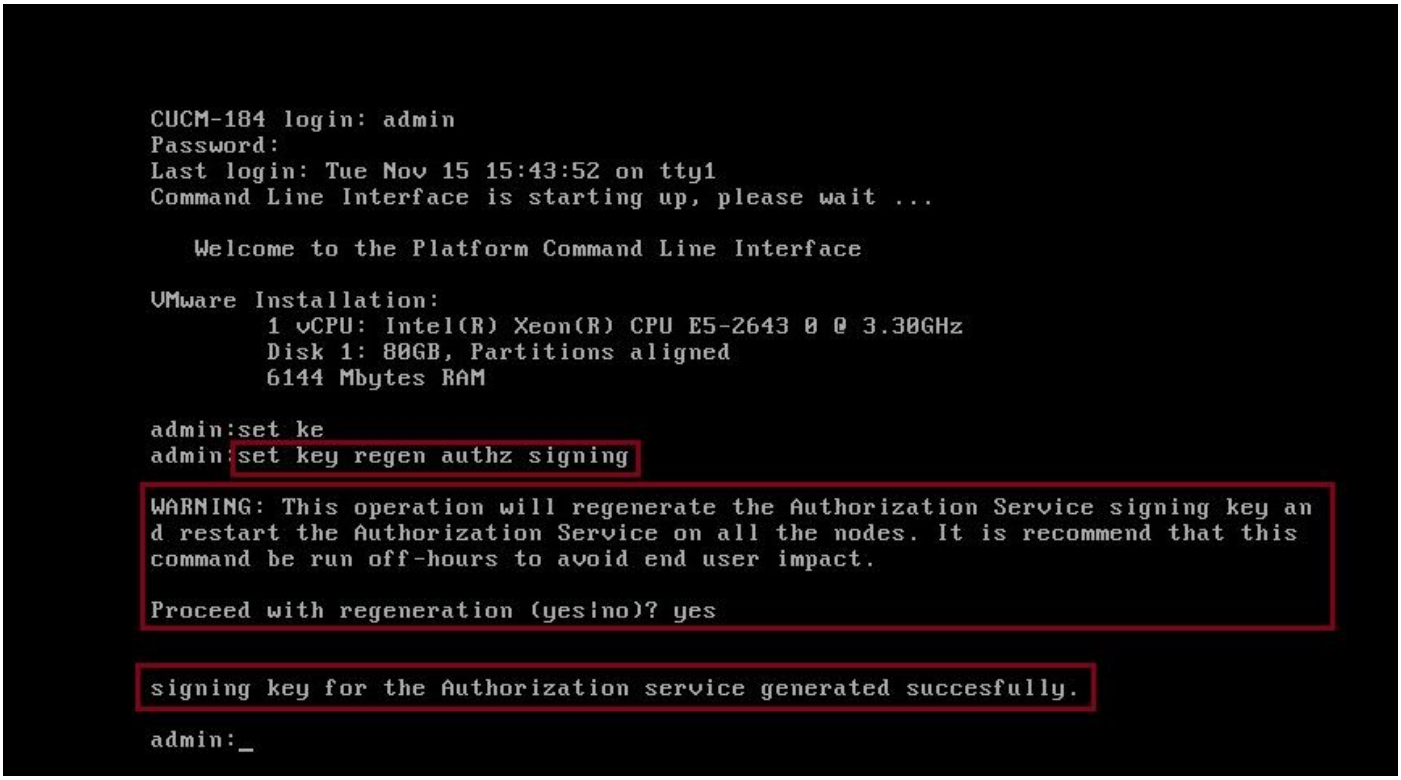
Clés de signature et de chiffrement

- La clé de signature est basée sur RSA, qui a une paire de clés publique/privée.
- La clé de chiffrement est une clé symétrique.
- Ces clés sont créées uniquement sur l'éditeur et sont distribuées sur tous les noeuds du cluster.
- La clé de signature et la clé de chiffrement peuvent être régénérées, avec l'utilisation des options répertoriées. Cependant, cela ne doit être fait que si l'administrateur estime que les clés ont été compromises. L'impact de la regénération de l'une ou l'autre de ces clés est que tous les jetons d'accès émis par le service AuthZ deviennent non valides.
- Les clés de signature peuvent être régénérées avec l'interface utilisateur et l'interface de ligne de commande.
- Les clés de chiffrement peuvent être régénérées uniquement avec l'interface de ligne de commande.

La régénération des certificats Authz (clé de signature) à partir de la page **Cisco Unified OS Administration** sur CUCM est illustrée dans l'image.



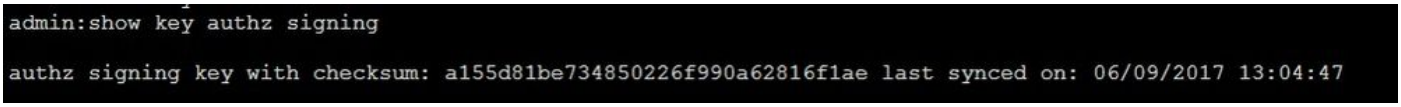
La régénération de la clé de signature Authz à l'aide de la commande CLI est illustrée dans l'image.



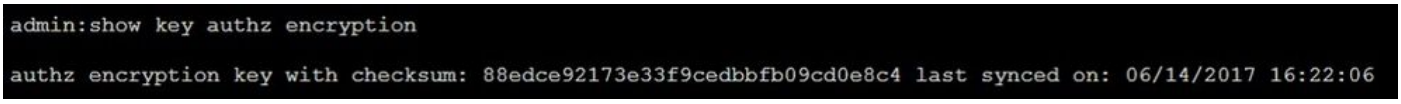
Admin peut afficher les clés de signature et de chiffrement d'authentification à l'aide de l'interface de ligne de commande. Le hachage de la clé s'affiche plutôt que la clé d'origine.

Les commandes permettant d'afficher les clés sont les suivantes :

Clé de signature : **show key authz sign** et comme illustré dans l'image.



Clé de chiffrement : **show key authz encryption** et comme illustré dans l'image.



Note: L'autorité de signature et l'autorité de chiffrement sont toujours différentes.

Vérification

Utilisez cette section pour confirmer que votre configuration fonctionne correctement.

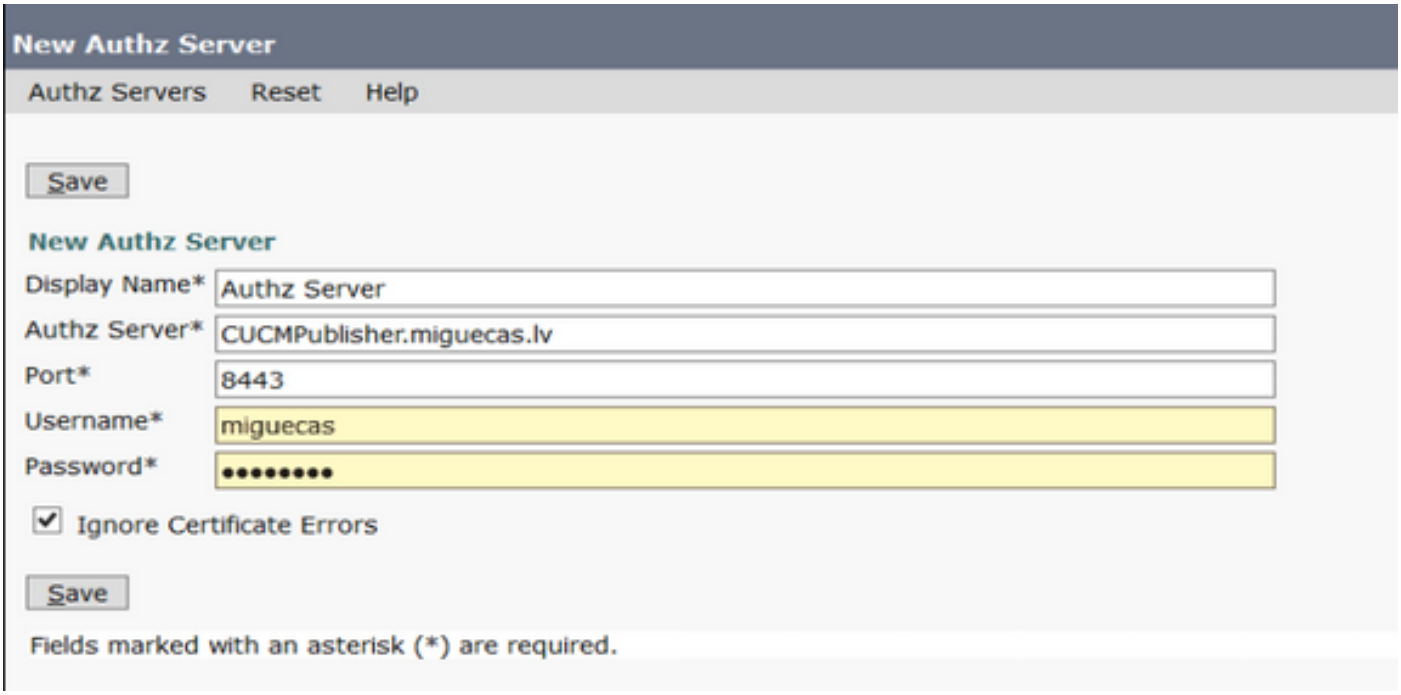
Lorsqu'il est prévu d'utiliser OAuth sur le serveur Cisco Unity Connection (CUC), l'administrateur réseau doit effectuer deux étapes.

Étape 1. Configurez le serveur Unity Connection pour récupérer les clés de signature et de chiffrement du jeton OAuth à partir du CUCM.

Étape 2. Activez OAuth Services sur le serveur CUC.

Remarque : pour récupérer les clés de signature et de chiffrement, Unity doit être configuré avec les détails de l'hôte CUCM et un compte utilisateur activé de l'accès AXL CUCM. Si ce paramètre n'est pas configuré, le serveur Unity ne peut pas récupérer le jeton OAuth à partir de CUCM et la connexion de messagerie vocale pour les utilisateurs ne peut pas être disponible.

Accédez à **Cisco Unity Connection Administration > System Settings > Authz Servers**



Dépannage

Cette section fournit les informations que vous pouvez utiliser pour dépanner votre configuration.

Note: Si OAuth est utilisé et que les utilisateurs de Cisco Jabber ne peuvent pas se connecter, vérifiez toujours les clés de signature et de chiffrement des serveurs CUCM et IM&P (Instant Messaging and Presence).

Les administrateurs réseau doivent **exécuter** ces deux commandes sur tous les noeuds CUCM et IM&P :

- **show key authz sign**
- **show key authz encryption**

Si les sorties d'authentification et d'authentification de chiffrement ne correspondent pas sur tous les noeuds, elles doivent être régénérées. Pour ce faire, ces deux commandes doivent être exécutées sur tous les noeuds CUCM et IM&P :

- **set key regen authz encryption**
- **set key regen authen**

Ensuite, le service **Cisco Tomcat** doit être redémarré sur tous les noeuds.

Outre l'incompatibilité des clés, cette ligne d'erreur se trouve dans les journaux Cisco Jabber :

```
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.
```

Les journaux des applications sso sont générés à ces emplacements :

- **fichier view activelog platform/log/ssoApp.log** Cette opération ne nécessite aucune configuration de suivi pour la collection de journaux. Chaque fois que l'application SSO est exécutée, de nouvelles entrées de journal sont générées dans le fichier ssoApp.log.
- Journaux SSOSP : **liste de fichiers activelog tomcat/logs/ssosp/log4j**
  Chaque fois que sso est activé, un nouveau fichier journal est créé à cet emplacement avec le nom **ssosp00XXX.log**. Toute autre opération SSO et toutes les opérations Oauth sont également connectées à ce fichier.
- Journaux de certificat : **liste de fichiers activelog platform/log/certMgmt*.log**
  Chaque fois que le certificat AuthZ est régénéré (interface utilisateur ou CLI), un nouveau fichier journal est généré pour cet événement.
  Pour la nouvelle génération de clé de chiffrement authz, un nouveau fichier journal est généré pour cet événement.

Informations connexes

[Déploiement d'OAuth avec la solution de collaboration Cisco version 12.0]