Implementación Y Resolución De Problemas De Flujo De Concesión De Código De Autorización - Mejora De OAuth: Soluciones de colaboración de Cisco 12.0

Contenido

Introducción

Este documento describe cómo el flujo de concesión de código de autorización se basa en el token de actualización para mejorar la experiencia del usuario de Jabber en varios dispositivos, especialmente para Jabber en Mobile.

Prerequisites

Requirements

Cisco recomienda que tenga conocimiento sobre estos temas:

- Cisco Unified Communications Manager (CUCM) 12.0 versión
- Inicio de sesión único (SSO)/SAML
- Cisco Jabber
- Microsoft ADFS
- Proveedor de identidad (IdP)

Para obtener más información sobre estos temas, consulte estos enlaces:

- Guía de implementación de SAML SSO para Cisco Unified Communications
- Ejemplo de Configuración de SSO de Unified Communications Manager SAML:
- Ejemplo de Configuración de AD FS Versión 2.0 para SAML SSO:

Componentes Utilizados

La información de este documento se basa en este software:

- Microsoft ADFS (IdP)
- LDAP Active Directory
- Cliente 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 tiene una red en vivo, asegúrese de entender el posible impacto de cualquier comando.

Antecedentes

A partir de hoy, el flujo SSO de Jabber con la infraestructura se basa en el flujo de concesiones implícito, en el que el servicio CUCM Authz asigna los tokens de acceso de corta duración.

Vencimiento del token de acceso posterior, CUCM redirige Jabber a IdP para la reautenticación.

Esto conlleva una mala experiencia para el usuario, especialmente con jabber en el móvil, donde se le pide al usuario que introduzca las credenciales con frecuencia.

La solución de rearquitectura de seguridad también propone el flujo de concesión de código de autorización (con el uso del enfoque de actualización de tokens (ampliable a terminales/otras aplicaciones de colaboración)) para la unificación del flujo de inicio de sesión de Jabber y terminales tanto en escenarios SSO como no SSO.

Características destacadas

- El flujo de concesión de código de autorización se basa en el token de actualización (extensible a los terminales/otras aplicaciones de colaboración) para mejorar la experiencia del usuario de Jabber en varios dispositivos, especialmente para Jabber en Mobile.
- Admite tokens autodefinidos firmados y cifrados de OAuth para permitir que diversas aplicaciones de colaboración validen y respondan a las solicitudes de recursos del cliente.
- Se conserva el modelo de flujo de concesión implícito que permite la compatibilidad con versiones anteriores. Esto también permite una ruta perfecta para otros clientes (como RTMT) que no se han trasladado al flujo de concesión de código de autorización.

Consideraciones importantes

- Implementación tal que el antiguo cliente Jabber pueda trabajar con el nuevo CUCM (ya que admite flujos de concesión de código de autorización y de concesión de permisos implícitos). Además, el nuevo jabber puede funcionar con el antiguo CUCM. Jabber puede determinar si CUCM admite el flujo de concesión de código de autorización y sólo si admite este modelo,

cambia y utiliza el flujo de concesión implícito.
- El servicio AuthZ se ejecuta en el servidor CUCM.
- AuthZ sólo admite flujo de concesión implícito. Esto significa que no se ha actualizado el token/token de acceso sin conexión. Cada vez que el cliente deseaba un nuevo token de acceso, el usuario necesita volver a autenticarse con el IdP.
- Los tokens de acceso se emitieron sólo si la implementación está habilitada para SSO. En este caso, las implementaciones que no son de SSO no funcionaron y los tokens de acceso no se utilizaron en todas las interfaces de forma uniforme.
- Los tokens de acceso no son independientes, sino que se conservan en la memoria del servidor que los emitió. Si CUCM1 emitió el token de acceso, sólo CUCM1 puede verificarlo. Si el cliente intenta acceder al servicio en CUCM2, CUCM2 necesita validar ese token en CUCM1. Retrasos de red (modo proxy).
- La experiencia del usuario en los clientes móviles es muy mala, ya que el usuario debe volver a introducir las credenciales en un teclado alfanumérico cuando el usuario se vuelve a autenticar con el IdP (normalmente, de 1 hora a 8 horas, lo que depende de varios factores).
- Los clientes que hablan con varias aplicaciones en varias interfaces necesitan mantener varias credenciales/bloques. No ofrece soporte para el mismo usuario para iniciar sesión desde 2 clientes similares. Por ejemplo, el usuario A inicia sesión desde instancias de Jabber que se ejecutan en 2 iPhones diferentes.
- AuthZ para admitir implementaciones SSO y no SSO.
- AuthZ para soportar flujo de concesión implícito + flujo de concesión de código de autorización. Como es **compatible con versiones anteriores,** permite a clientes como **RTMT** continuar trabajando hasta que se adapten.
- Con el flujo de concesión de código de autorización, AuthZ emite el token de acceso y el token de actualización. El token de actualización se puede utilizar para obtener otro token de acceso sin necesidad de autenticación.
- Los tokens de acceso son independientes, están firmados y cifrados y utilizan el estándar JWT (tokens web JSON) (compatible con RFC).
- Las claves de firma y cifrado son comunes al clúster. Cualquier servidor del clúster puede verificar el token de acceso. No hay necesidad de mantener la memoria.
- el servicio que se ejecuta en CUCM 12.0 es el servidor de autenticación centralizado en el clúster.
- Los tokens de actualización se almacenan en la base de datos (DB). El administrador debe poder revocarlo, si es necesario. La revocación se basa en ID de usuario o ID de usuario y ID de cliente.
- Los tokens de acceso firmados permiten que diferentes productos validen tokens de acceso sin necesidad de almacenarlos. Token de acceso configurable y tiempos de vida del token de actualización (1 hora y 60 días, respectivamente).
- El formato JWT está alineado con Spark, lo que permite sinergias en el futuro con los servicios híbridos de Spark.
- Compatibilidad con el mismo usuario que inicia sesión desde 2 dispositivos similares. Por ejemplo: El usuario A puede iniciar sesión desde instancias de Jabber que se ejecutan en 2 iPhones diferentes.

Elementos del flujo de concesión de código de autorización

- Servidor Auth Z

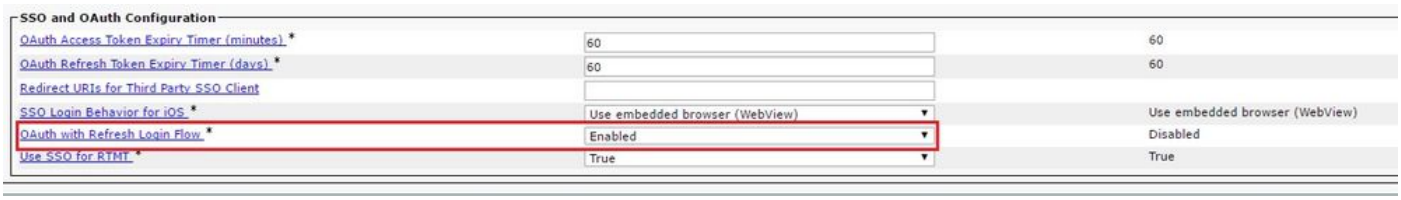
- Claves de cifrado
- Claves de firma
- Actualizar tokens

Configurar

Esta función no está activada de forma predeterminada.

Paso 1. Para habilitar esta función, navegue hasta **System > Enterprise Parameters**.

Paso 2. Establezca el parámetro **OAuth con Refresh Login Flow** en Enabled como se muestra en la imagen.



- El token de acceso se firma y se cifra. La firma y la clave de cifrado son comunes al clúster. Esto significa que cualquier nodo del clúster puede validar el token de acceso.
- El token de acceso está en formato JWT (RFC 7519).
- Los tokens de acceso reutilizan el parámetro de empresa (temporizador de caducidad de token de acceso OAuth), que se aplica tanto a los formatos de token antiguos como a los nuevos.
- Valor predeterminado: 60 min.
- Valor mínimo: 1 min
- Valor máximo: 1440 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
```

Se emite un nuevo token de acceso cada vez que el cliente solicita uno. El antiguo sigue siendo válido mientras:

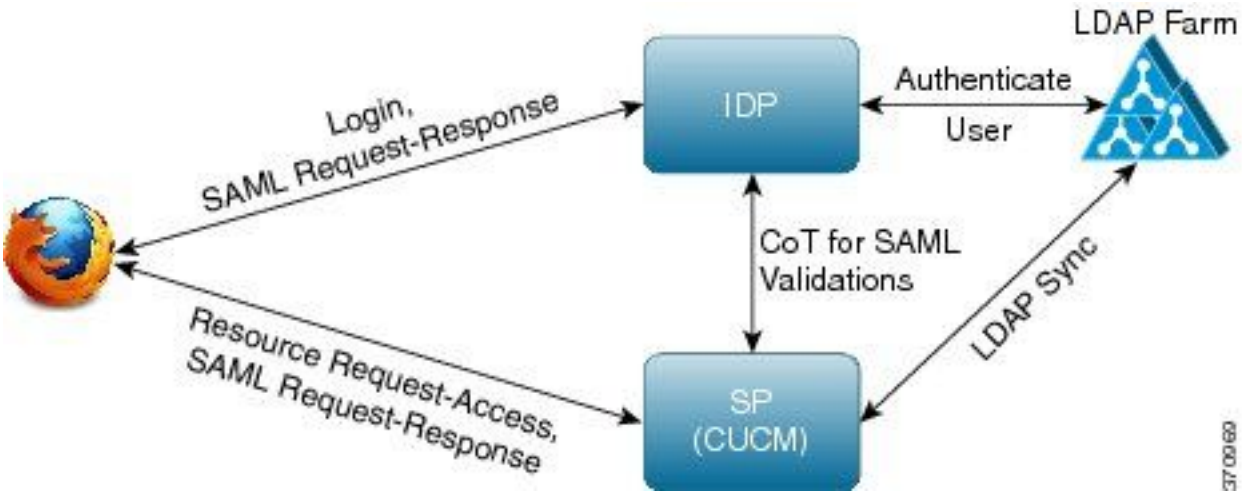
- Las claves de firma/cifrado no han cambiado
- La validez (almacenada dentro del token) se rompe.
- JSON web-tokens : consta de tres partes, separadas por puntos, que son: Encabezado, carga útil y firma.

Ejemplo de token de acceso:

- Al principio del token que se resalta en negrita se encuentra el encabezado.
- La parte central es la carga útil.
- Al final, si el token está resaltado en negrita, entonces es la firma.

Diagrama de la red

A continuación se ofrece una descripción general de alto nivel del flujo de llamadas involucrado:



Actualizar tokens

- El token de actualización está firmado.
- El token de actualización se almacena en la tabla **actutokendetails** de la base de datos como un valor hash propio. Esto es para evitar la replicación por parte de la base de datos, ya que alguien puede elegirla. Para revisar la tabla, puede ejecutar:
  `run sql select * from refreshtokendetails`
  O con una fecha de validez legible:

```
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
```

> **Advertencia:** El token de actualización se vacía de la base de datos cuando caduca la validez. El subproceso del temporizador se ejecuta a las 2 am cada día (no se puede configurar a través de la interfaz de usuario, pero se puede modificar a través de una cuenta de soporte remoto). Si la tabla tiene un gran número de tokens de acceso, no son válidos y deben eliminarse. Esto puede causar un pico de CPU.

Sample refresh token:


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


Revocar los tokens de actualización

Admin tiene la capacidad de revocar todos los tokens de actualización para un usuario o tokens de actualización sólo para dispositivos para un usuario a través de **userID** o **userID** y **ClientID**.

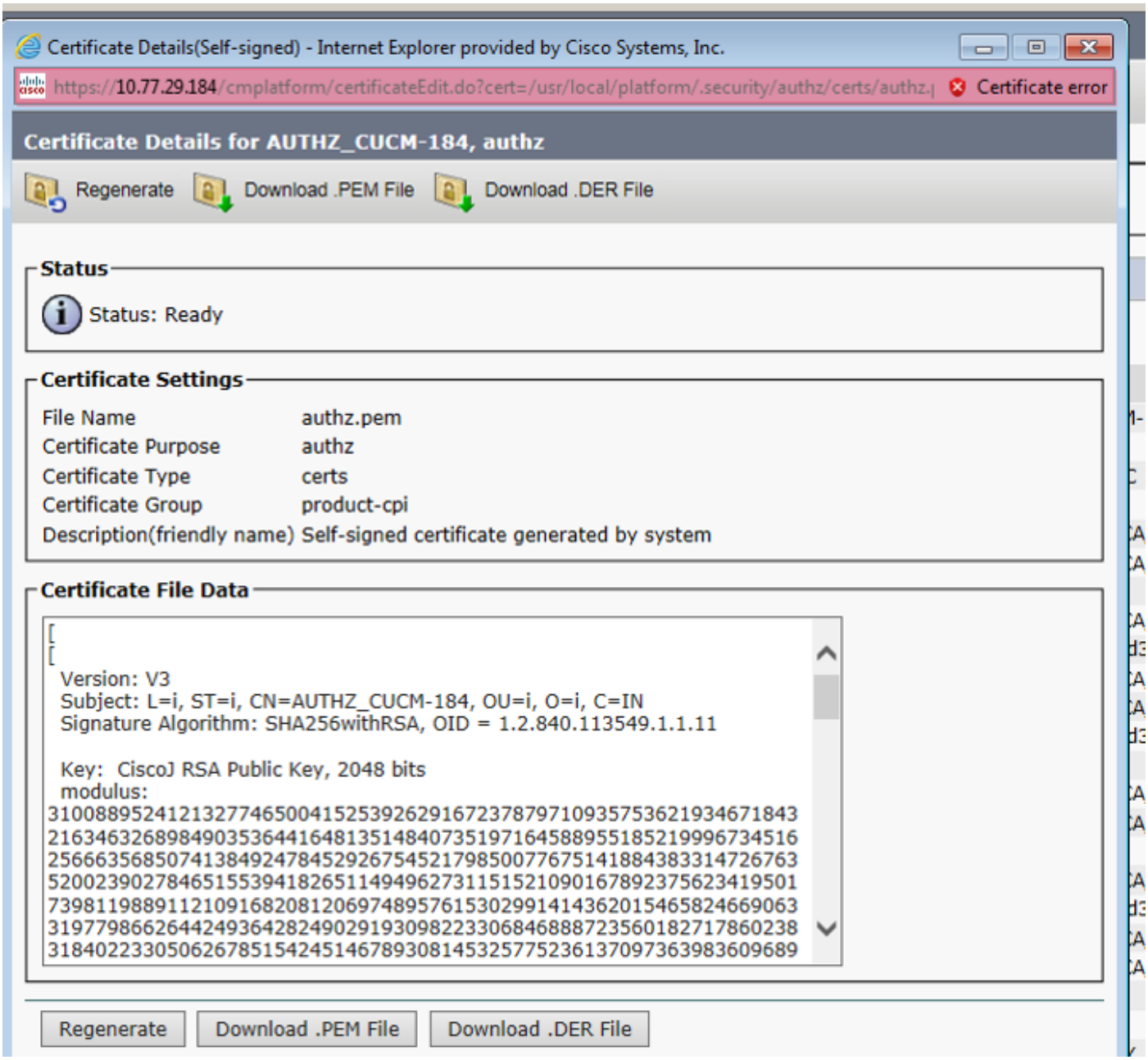
Para revocar los RT basados en dispositivos para un usuario:

- revoque RT para el usuario xyz y el dispositivo identificado por client_id abc.
- https://cucm-193:8443/ssosp/token/revoke?user_id=xyz&client_id=abc

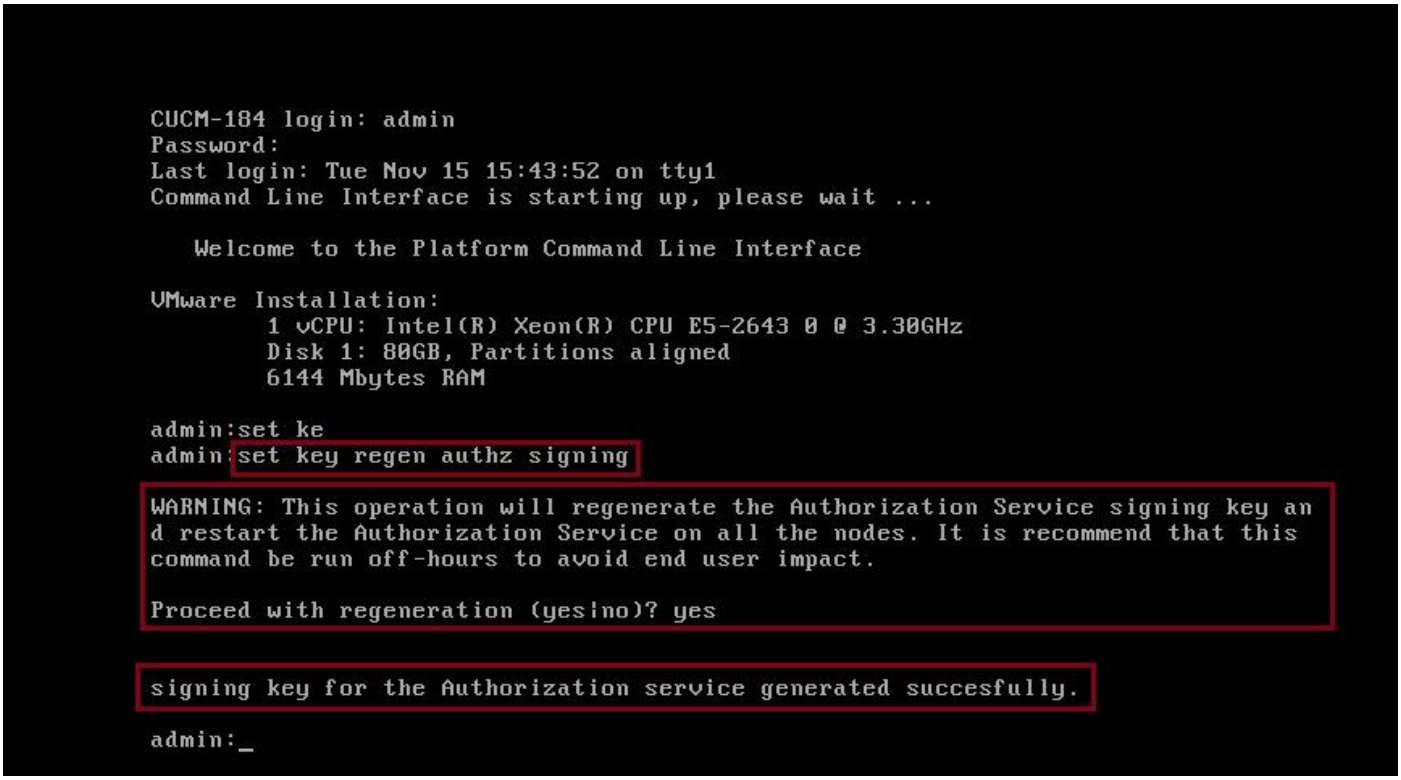
Claves de firma y cifrado

- La clave de firma se basa en RSA, que tiene un par de claves pública y privada.
- La clave de cifrado es una clave simétrica.
- Estas claves se crean sólo en el editor y se distribuyen en todos los nodos del clúster.
- Tanto la clave de firma como la clave de cifrado se pueden volver a generar, con el uso de las opciones enumeradas. Sin embargo, esto sólo debe hacerse si el administrador cree que las claves se han visto comprometidas. El impacto de la regeneración de cualquiera de estas claves es que todos los tokens de acceso emitidos por el servicio AuthZ se vuelven inválidos.
- Las llaves de firma se pueden volver a generar con la interfaz de usuario y la CLI.
- Las claves de cifrado sólo se pueden regenerar con CLI.

La regeneración de certificados Authz (clave de firma) de la página **Administración de Cisco Unified OS** en CUCM es como se muestra en la imagen.



La regeneración de la llave de firma Authz con el uso del comando CLI es como se muestra en la imagen.



El administrador puede mostrar las claves de firma y cifrado de autenticación con el uso de CLI. Se muestra el hash de la clave en lugar de la clave original.

Las teclas de comandos para mostrar son:

Clave de firma: **mostrar firma de autenticación de clave** y como se muestra en la imagen.

```
admin:show key authz signing

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

Clave de cifrado: **mostrar cifrado de autenticación de clave** y como se muestra en la imagen.

```
admin:show key authz encryption

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

Nota: La firma authz y la autenticación de cifrado son siempre diferentes.

Verificación

Utilize esta sección para confirmar que su configuración funcione correctamente.

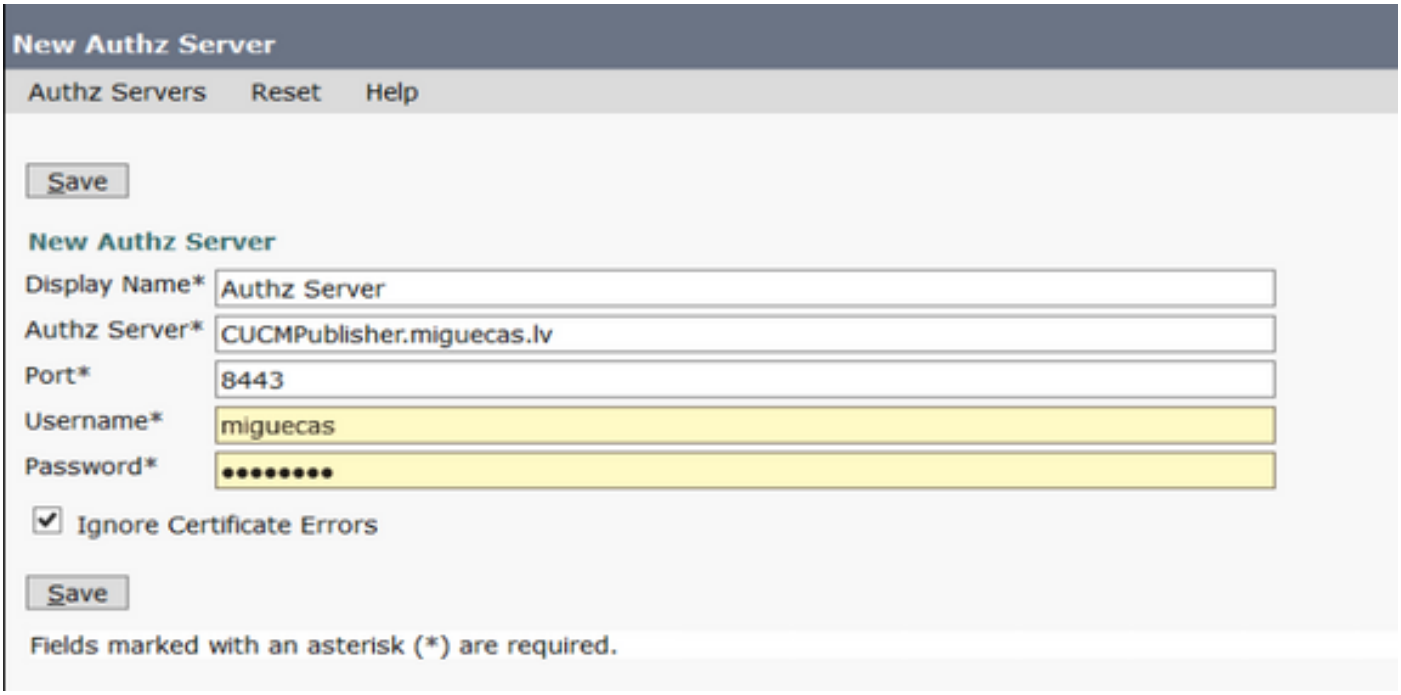
Cuando se pretende utilizar OAuth en el servidor de Cisco Unity Connection (CUC), el administrador de la red debe realizar dos pasos.

Paso 1. Configure el Servidor de Unity Connection para obtener la firma OAuth Token y las claves de cifrado de CUCM.

Paso 2. Habilite los servicios OAuth en el servidor CUC.

Nota: Para obtener las claves de firma y cifrado, Unity debe configurarse con los detalles del host de CUCM y una cuenta de usuario habilitada del acceso AXL de CUCM. Si esto no se configura, el servidor de Unity no puede recuperar el token OAuth de CUCM y el inicio de sesión del correo de voz para los usuarios no puede estar disponible.

Vaya a **Administración de Cisco Unity Connection > Configuración del sistema > Servidores de autenticación**



Troubleshoot

Esta sección proporciona la información que puede utilizar para resolver problemas de su configuración.

Nota: Si se utiliza OAuth y los usuarios de Cisco Jabber no pueden iniciar sesión, revise siempre las claves de firma y cifrado de los servidores de CUCM y mensajería instantánea y presencia (IM&P).

Los administradores de red necesitan **ejecutar** estos dos comandos en todos los nodos CUCM y IM&P:

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

Si las salidas de autenticación de firma y autenticación de cifrado no coinciden en todos los nodos, es necesario regenerarlas. Para realizar esto, estos dos comandos deben ejecutarse en todos los nodos CUCM e IM&P:

- **set key regen authz encryption**
- **set key regen authz sign**

Después, el servicio **Cisco Tomcat** debe reiniciarse en todos los nodos.

Junto con la discordancia de las claves, esta línea de error se puede encontrar en los registros de

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.
```

Los registros de la aplicación sso se generan en estas ubicaciones:

- **file view activelog platform/log/ssoApp.log** Esto no requiere ninguna configuración de seguimiento para la recopilación de registros. Cada vez que se realiza la operación de la aplicación SSO, se generan nuevas entradas de registro en el archivo ssoApp.log.
- Registros SSOSP: **file list activelog tomcat/logs/ssosp/log4j**
  Cada vez que se habilita sso, se crea un nuevo archivo de registro en esta ubicación con el nombre **ssosp00XXX.log**. Cualquier otra operación SSO y todas las operaciones de Oauth también se registran en este archivo.
- Registros de certificados: **file list activelog platform/log/certMgmt*.log**
  Cada vez que se regenera el certificado AuthZ (UI o CLI), se genera un nuevo archivo de registro para este evento.
  Para la regeneración de la clave de cifrado authz, se genera un nuevo archivo de registro para este evento.

Información Relacionada

Implementación de OAuth con la versión 12.0 de la solución de colaboración de Cisco