Implantar E Solucionar Problemas De Fluxo De Concessão De Código De Autorização - Aprimoramento De OAuth: Soluções Cisco Collaboration 12.0

Contents

Introduction

Este documento descreve como o fluxo de concessão de código de autorização é baseado no token de atualização para melhorar a experiência do usuário Jabber em vários dispositivos, especialmente para Jabber no celular.

Prerequisites

Requirements

A Cisco recomenda que você tenha conhecimento destes tópicos:

- Cisco Unified Communications Manager (CUCM) versão 12.0
- Login único (SSO)/SAML
- Cisco Jabber
- Microsoft ADFS
- Provedor de identidade (IdP)

Para obter mais informações sobre esses tópicos, consulte estes links:

- Guia de implantação do SSO SAML para Cisco Unified Communications
- Exemplo de configuração do SSO SAML do Unified Communications Manager:
- Exemplo de configuração de AD FS versão 2.0 para SAML SSO:

Componentes Utilizados

As informações neste documento são baseadas neste software:

- Microsoft ADFS (IdP)
- Diretório ativo LDAP
- 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. Se a rede estiver ativa, certifique-se de que você entenda o impacto potencial de qualquer comando.

Informações de Apoio

A partir de hoje, o fluxo do Jabber SSO com infraestrutura é baseado no fluxo de concessão implícito, no qual o serviço CUCM Authz aloca os tokens de acesso de curta duração.

Após a expiração do token de acesso, o CUCM redireciona o Jabber para IdP para reautenticação.

Isso leva a uma má experiência do usuário, especialmente com o jabber no celular, onde o usuário é solicitado a inserir credenciais frequentemente.

A solução de rearquitetura de segurança também propõe o fluxo de concessão de código de autorização (com o uso da abordagem de tokens de atualização (extensível a endpoints/outros aplicativos de colaboração) para unificação do fluxo de login do Jabber e do endpoint para cenários SSO e não SSO.

Destaques dos recursos

- O fluxo de concessão de código de autorização é baseado no token de atualização (extensível a endpoints/outros aplicativos de colaboração) para melhorar a experiência do usuário Jabber em vários dispositivos, especialmente para Jabber no celular.
- Suporta Tokens OAuth Autocontidos Assinados e Criptografados para permitir que vários aplicativos de colaboração validem e respondam às solicitações de recursos do cliente.
- O modelo de fluxo de concessão implícito é mantido, permitindo compatibilidade com versões anteriores. Isso também permite um caminho perfeito para outros clientes (como RTMT) que não mudaram para o fluxo de concessão de código de autorização.

Considerações importantes

- Implementação de modo que o antigo cliente Jabber possa trabalhar com o novo CUCM (já que ele suporta fluxos implícitos de concessão de concessão e de código de autorização). Além disso, o novo jabber pode trabalhar com o antigo CUCM. O Jabber pode determinar se

o CUCM suporta o fluxo de concessão de código de autorização e somente se suportar esse modelo, ele comuta e usa o fluxo de concessão implícito.
- O serviço AuthZ é executado no servidor CUCM.
- AuthZ suporta apenas fluxo de concessão implícito. Isso significa que não há token de acesso token/offline de atualização. Cada vez que o cliente queria um novo token de Acesso, o usuário precisa autenticar novamente com o IdP.
- Tokens de acesso foram emitidos somente se sua implantação estiver habilitada para SSO. As implantações não-SSO não funcionaram nesse caso e os tokens de acesso não foram usados em todas as interfaces de forma consistente.
- Os Tokens de acesso não são independentes, mas sim retidos na memória do servidor que os emitiu. Se o CUCM1 tiver emitido o token de acesso, ele poderá ser verificado somente pelo CUCM1. Se o cliente tentar acessar o serviço no CUCM2, o CUCM2 precisará validar esse token no CUCM1. Atrasos na rede (modo proxy).
- A experiência do usuário em clientes móveis é muito ruim, pois o usuário precisa digitar novamente as credenciais em um teclado alfanumérico quando o usuário se autentica novamente com o IdP (normalmente executado de 1 hora a 8 horas, o que depende de vários fatores).
- Os clientes que falam com vários aplicativos em várias interfaces precisam manter várias credenciais/blocos. Não há suporte direto para o mesmo login de usuário de 2 clientes semelhantes. Por exemplo, o usuário A faz login em instâncias do jabber que são executadas em dois iPhones diferentes.
- AuthZ para suportar implantações SSO e não SSO.
- AuthZ para suportar fluxo de concessão implícito + fluxo de concessão de código de autorização. Como é **compatível com versões anteriores,** permite que clientes como a **RTMT** continuem trabalhando até se adaptarem.
- Com o fluxo de concessão de código de autorização, a AuthZ emite token de acesso e token de atualização. O token de atualização pode ser usado para obter outro token de acesso sem a necessidade de autenticação.
- Os tokens de acesso são independentes, assinados e criptografados e usam o padrão JWT (JSON web tokens) (compatível com RFC).
- As chaves de assinatura e criptografia são comuns ao cluster. Qualquer servidor no cluster pode verificar o token de acesso. Não há necessidade de manutenção na memória.
- o serviço que é executado no CUCM 12.0 é o Servidor de Autenticação centralizado no cluster.
- Os tokens de atualização são armazenados no banco de dados (DB). O administrador precisa ser capaz de revogá-lo, se necessário. A revogação baseia-se na ID de usuário ou na ID de cliente.
- Os tokens de acesso assinados permitem que diferentes produtos validem tokens de acesso sem a necessidade de armazená-los. Token de acesso configurável e tempos de vida do token de atualização (padrão de 1 hora e 60 dias, respectivamente).
- O formato JWT está alinhado com o Spark, o que permite sinergias no futuro com os serviços Spark Hybrid.
- Suporte para o mesmo usuário que faz login a partir de 2 dispositivos semelhantes. Por exemplo: O Usuário A pode fazer login a partir de instâncias do jabber executadas em dois iPhones diferentes.

Fluxo de concessão de código de autorização de elementos

- Servidor Z Auth
- Chaves de criptografia
- Chaves de assinatura
- Atualizar tokens

Configurar

Este recurso não está ativado por padrão.

Etapa 1. Para habilitar esse recurso, navegue para **System > Enterprise Parameters**.

Etapa 2. Defina o parâmetro **OAuth com Refresh Login Flow** como Enabled (Habilitado), como mostrado na imagem.

- O token de acesso é assinado e criptografado. A chave de assinatura e criptografia é comum ao cluster. Isso significa que qualquer nó no cluster pode validar o token de acesso.
- O token de acesso está no formato JWT (RFC 7519).
- Os tokens de acesso reutilizam o parâmetro da empresa (temporizador de expiração de token de acesso OAuth), que é aplicável para os formatos de token antigo e novo.
- Valor padrão - 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
```

Um novo token de acesso é emitido cada vez que o cliente solicita um. O antigo continua válido desde que:

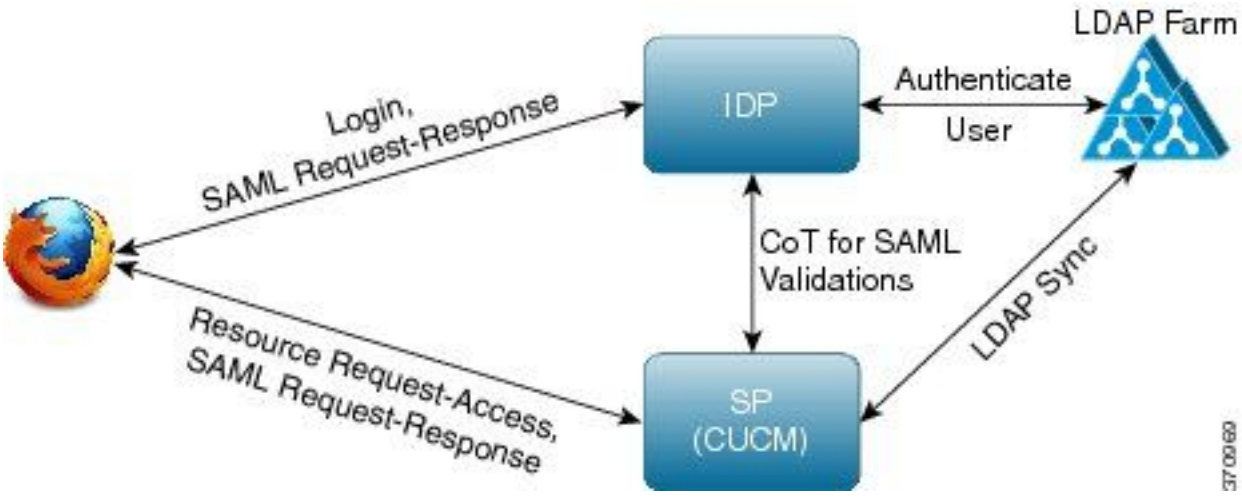
- As chaves de assinatura/criptografia não foram alteradas
- Validade (armazenada dentro do token) interrompe.
- Web-tokens JSON: Consistem em três partes, separadas por pontos, que são: Cabeçalho, payload e assinatura.

Exemplo de token de acesso:

- No início do token destacado em negrito está o cabeçalho.
- A parte do meio é o Payload.
- No final, se o token estiver realçado em negrito, ele será a Assinatura.

Diagrama de Rede

Aqui está uma visão geral de alto nível do fluxo de chamadas envolvido:



Atualizar tokens

- O token de atualização está assinado.
- O token de atualização é armazenado na tabela de **detalhes** de **atualização** no banco de dados como um valor de hash. Isso evita a replicação pelo banco de dados, pois ele pode ser escolhido por alguém. Para revisar a tabela, você pode executar:
  `run sql select * from refreshtokendetails`
  Ou com uma data de validade legível:

```
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
```

aviso: O token de atualização é descarregado do banco de dados quando a validade expira. O thread do temporizador é executado às 2 da manhã do dia (não configurável por IU, mas pode ser modificado por conta de suporte remoto). Se a tabela tiver um grande número de tokens de acesso, que são inválidos e precisam ser eliminados. Isso pode causar um pico na CPU.

Sample refresh token:


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


Revogar Tokens de Atualização

O administrador tem a capacidade de revogar todos os tokens de atualização para um usuário ou tokens de atualização somente de dispositivo para um usuário por meio de **userID** ou **userID** e **ClientID**.

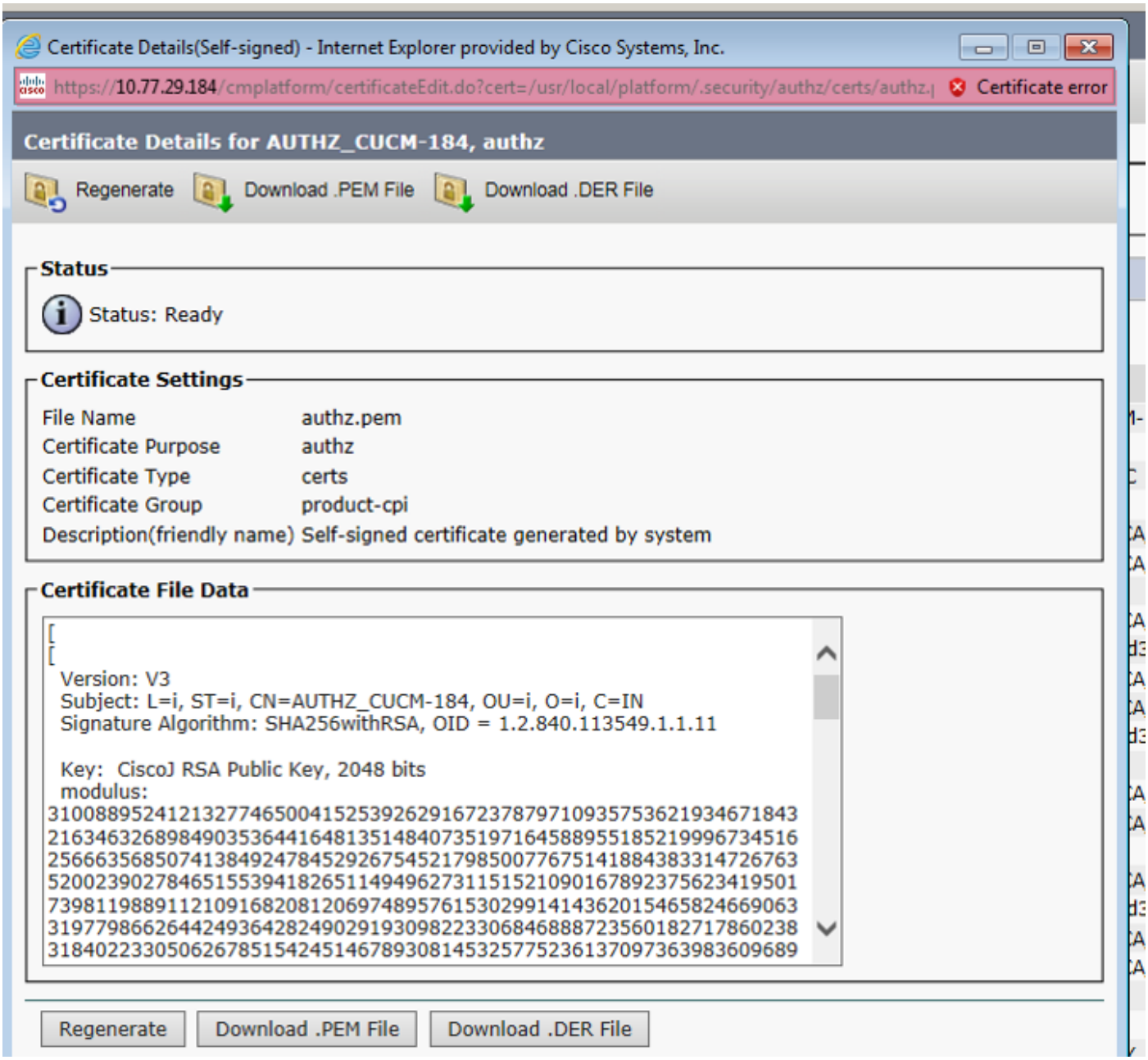
Para revogar RTs baseadas em dispositivo para um usuário:

- revogar RT para o usuário xyz e dispositivo identificado pelo client_id abc.
- https://cucm-193:8443/ssosp/token/revoke?user_id=xyz&client_id=abc

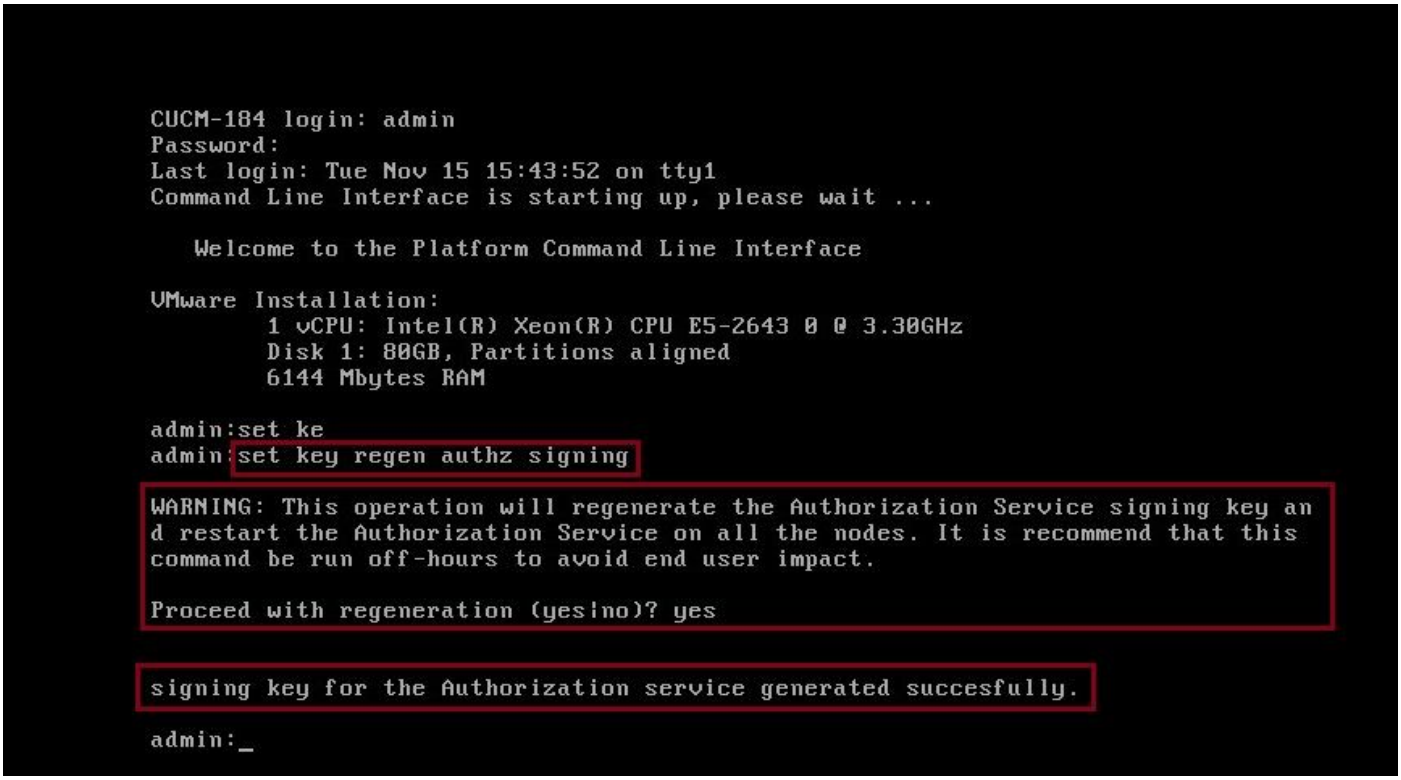
Chaves de assinatura e criptografia

- A chave de assinatura é baseada em RSA, que tem par de chaves públicas/privadas.
- A chave de criptografia é uma chave simétrica.
- Essas chaves são criadas apenas no editor e distribuídas por todos os nós no cluster.
- A chave de assinatura e a chave de criptografia podem ser regeradas, com o uso das opções listadas. No entanto, isso só deve ser feito se o administrador acreditar que as chaves foram comprometidas. O impacto da regeração dessas chaves é que todos os tokens de acesso emitidos pelo serviço AuthZ se tornam inválidos.
- As chaves de assinatura podem ser regeradas com a IU e a CLI.
- As chaves de criptografia podem ser regeneradas somente com CLI.

A regeneração de certificados do Authz (chave de assinatura) na página **Cisco Unified OS Administration** no CUCM é como mostrado na imagem.



A regeneração da chave de assinatura Auz com o uso do comando CLI é como mostrado na imagem.



O administrador pode exibir chaves de criptografia e de assinatura automática com o uso de CLI. O hash da chave é exibido em vez da chave original.

Os comandos para exibir chaves são:

Chave de assinatura: **show key authz signed** e como mostrado na imagem.

```
admin:show key authz signing

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

Chave de criptografia: **show key authz encryption** e como mostrado na imagem.

```
admin:show key authz encryption

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

Note: A autenticação e a autenticação de criptografia são sempre diferentes.

Verificar

Use esta seção para confirmar se a sua configuração funciona corretamente.

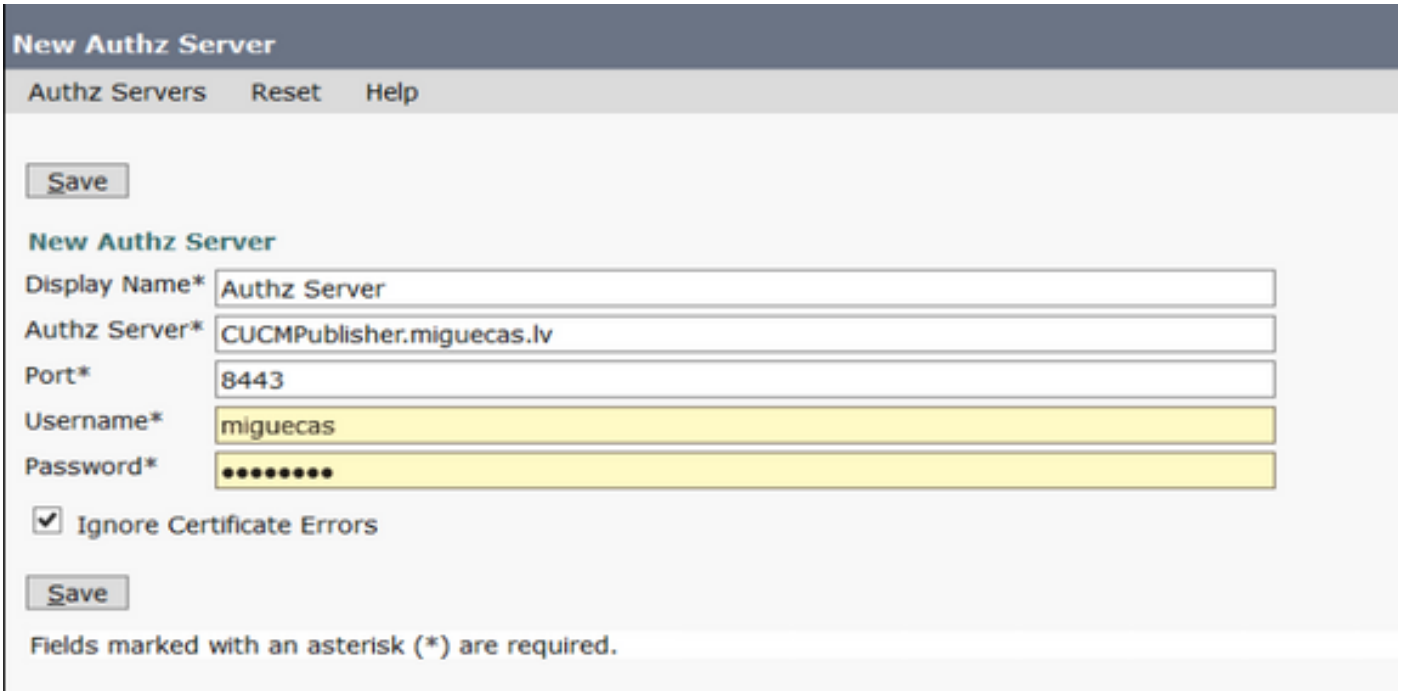
Quando a finalidade é usar OAuth no servidor do Cisco Unity Connection (CUC), o administrador da rede deve executar duas etapas.

Etapa 1. Configure o Unity Connection Server para buscar as chaves de criptografia e assinatura de token OAuth do CUCM.

Etapa 2. Ative os serviços OAuth no servidor CUC.

Observação: para buscar as chaves de assinatura e criptografia, o Unity deve ser configurado com os detalhes do host do CUCM e uma conta de usuário habilitada do CUCM AXL Access. Se isso não estiver configurado, o Unity Server não poderá recuperar o Token OAuth do CUCM e o correio de voz conectado para os usuários não poderá estar disponível.

Navegue até **Cisco Unity Connection Administration > System Settings > Authz Servers**



Troubleshoot

Esta seção fornece as informações que você pode usar para solucionar problemas de sua configuração.

Note: Se o OAuth for usado e os usuários do Cisco Jabber não conseguirem fazer login, revise sempre as chaves de assinatura e criptografia dos servidores CUCM e Mensagens instantâneas e Presença (IM&P).

Os administradores de rede precisam **executar** estes dois comandos em todos os nós CUCM e IM&P:

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

Se as saídas de autenticação e autenticação de criptografia não coincidirem em todos os nós, elas precisam ser regeneradas. Para fazer isso, esses dois comandos precisam ser executados em todos os nós CUCM e IM&P:

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

Depois, o serviço **Cisco Tomcat** precisa ser reiniciado em todos os nós.

Juntamente com a incompatibilidade das chaves, esta linha de erro pode ser encontrada nos

registros do 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.
```

Os registros de aplicativos sso são gerados nestes locais:

- **file view activelog platform/log/ssoApp.log** Isso não exige nenhuma configuração de rastreamento para a coleta de logs. Toda vez que a operação do aplicativo SSO é concluída, novas entradas de log são geradas no arquivo ssoApp.log.
- Logs SSOSP: **lista de arquivos activelog tomcat/logs/ssosp/log4j**
  Sempre que o sso é ativado, é criado um novo ficheiro de registro nesta localização com o nome **ssosp00XXX.log**. Qualquer outra operação SSO e todas as operações Oauth também estão registradas neste arquivo.
- Registros de certificado: **lista de arquivos activelog platform/log/certMgmt*.log**
  Cada vez que o certificado AuthZ é regenerado (UI ou CLI), um novo arquivo de log é gerado para esse evento.
  Para a regeração da chave de criptografia authz, um novo arquivo de log é gerado para este evento.

Informações Relacionadas

Implantação da OAuth com a Cisco Collaboration Solution versão 12.0