Configure and Troubleshoot the WebApp SSO on CMS

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Background

Configure

Network Diagram

ADFS Installation and Initial setup

Map CMS Users to Identity Provider (IdP)

Create Webbridge Metadata XML for IdP

Import Metadata for Webbridge into Identity Provider (IdP)

Create Claim Rules for the Webbridge Service on the IdP

Create SSO Archive ZIP file for Webbridge:

- Obtain and configure the idp_config.xml
- Create the config.jsonFile with Contents
- Set the sso_sign.key (OPTIONAL)
- Set the sso_encrypt.key (OPTIONAL)
- Creating the SSO ZIP file
- Upload the SSO Zip file(s) to Webbridge

Common Access Card (CAC)

Testing SSO Log in via WebApp

Troubleshooting

- Basic Troubleshooting
- Microsoft ADFS failure codes
- Failed to obtain authenicationID
- No assertion passed/matched in validation
- Sign in Failed on Web App:
  - Scenario 1:
  - Scenario 2:
  - Scenario 3:
- Username is not Recognized
  - Scenario 1:
  - Scenario 2:
- Webbridge log showing working log in example. Example generated using ?trace=true in the join URL:

Frequently Asked Questions (FAQ)

- Can the Webapp SSO JWT be extended?
- Do I need to re-authenticate to WebApp if I close my browser?
- How are multiple domains supported in WebApp SSO?

Related Information

Introduction

This document describes the how to configure and troubleshoot the Cisco Meeting Server (CMS) Web App implementation of Single Sign On (SSO).

Prerequisites

Requirements

Cisco recommends you have knowledge of these topics:

- CMS Callbridge version 3.1 or later
- CMS Webbridge version 3.1 or later
- Active Directory Server
- Identify Provider (IdP)

Components Used

The information in this document is based on these software and hardware versions:

- CMS Callbridge version 3.2
- CMS Webbridge version 3.2
- Microsoft Active Directory Windows Server 2012 R2
- Microsoft ADFS 3.0 Windows Server 2012 R2

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

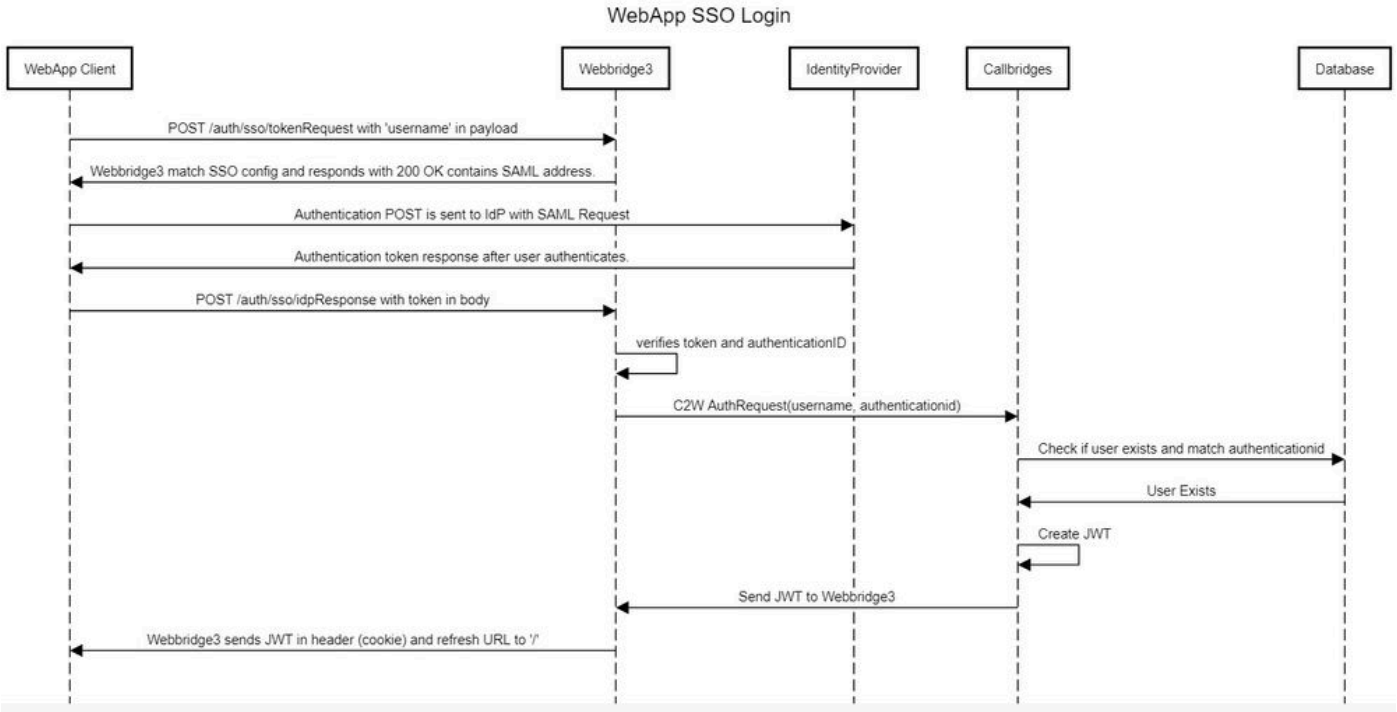
CMS 3.1 and later introduced the capability for users to sign in using an SSO without the need to enter their password every time the user logs in, because a single session is created with the Identify provider.
This feature is using the Security Assertion Markup Language (SAML) version 2.0 as the SSO Mechanism.

Note: CMS only supports HTTP-POST bindings in the SAML 2.0 and rejects any Identify Provider with no HTTP-POST bindings available.

Note: When SSO is enabled, basic LDAP authentication is no longer possible.

Configure

Network Diagram



ADFS Installation and Initial setup

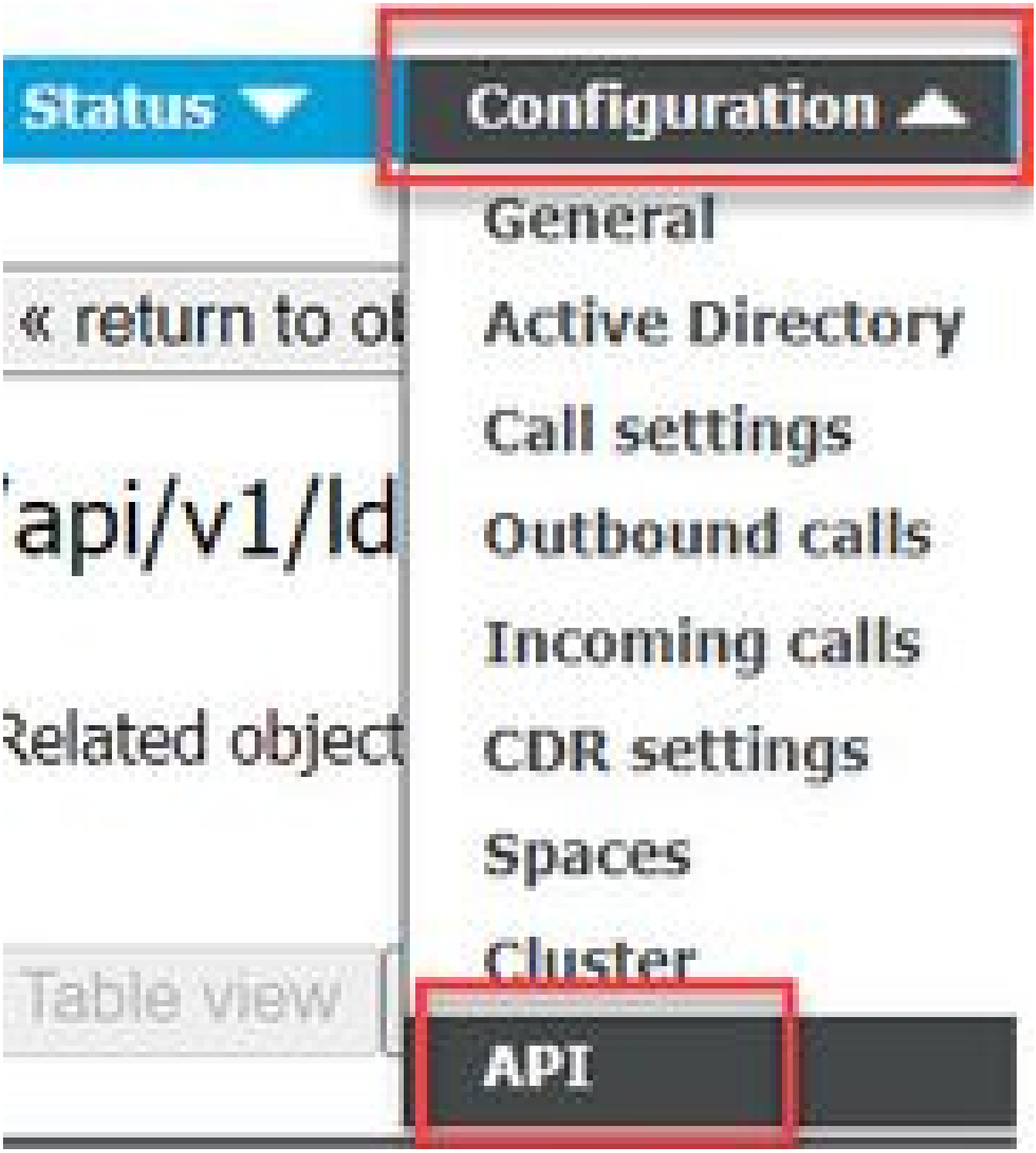
This deployment scenario uses Microsoft Active Directory Federation Services (ADFS) as the Identity Provider (IdP) and therefore, it is suggested to have an ADFS (or intended IdP) installed and running prior to this configuration.

Map CMS Users to Identity Provider (IdP)

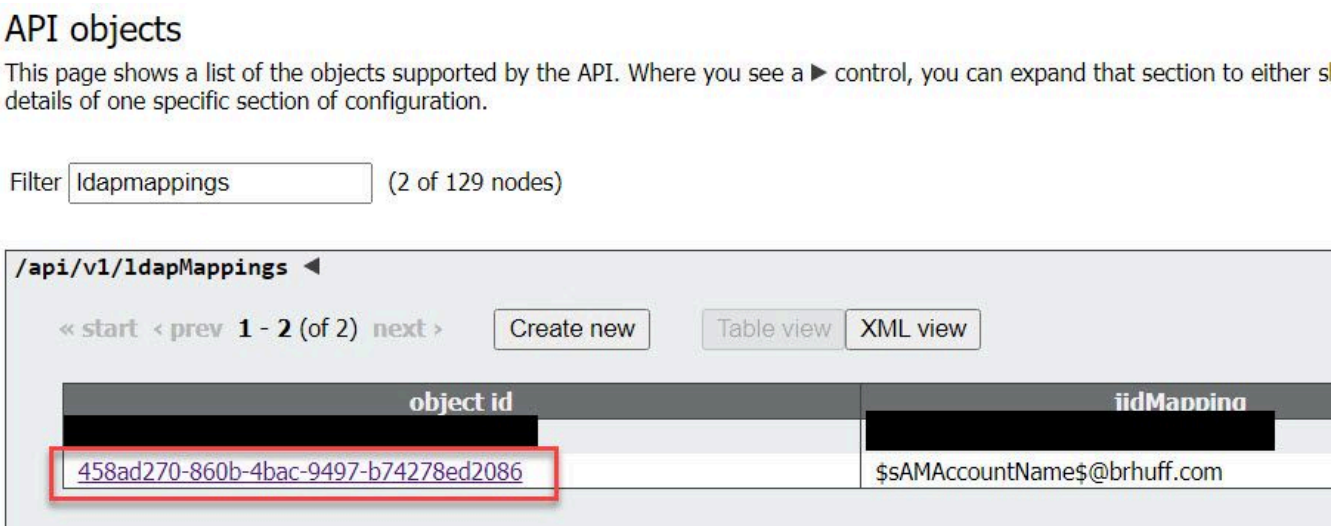
In order to have users get valid authentication, they must to be mapped in the Application Programming Interface (API) for a correlating field provided by IdP. The option used for this is the **authenicationIdMapping** in the **ldapMapping** of the API.

1. Navigate to **Configuration > API** on the CMS Web Admin GUI

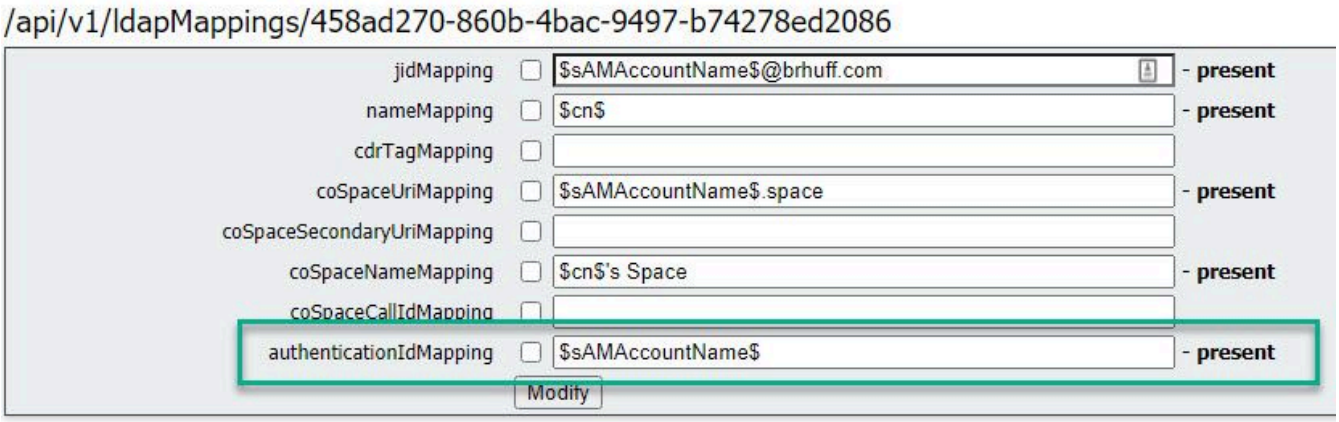
    Co



2. Locate existing (or creating a new) LDAP Mapping under **api/v1/ldapMappings/<GUID-of-Ldap-Mapping>**.



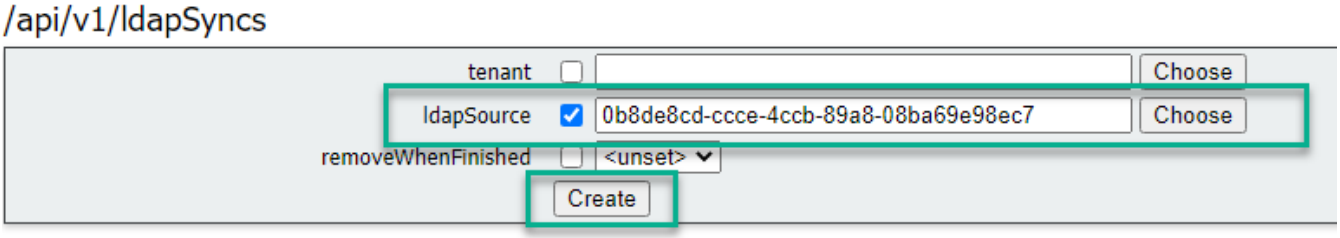
3. In the **ldapMapping** object selected, update the **authenticationIdMapping** to the LDAP attribute that is passed from the IdP. In the example, the option **$sAMAccountName**is used as the LDAP attribute for mapping.



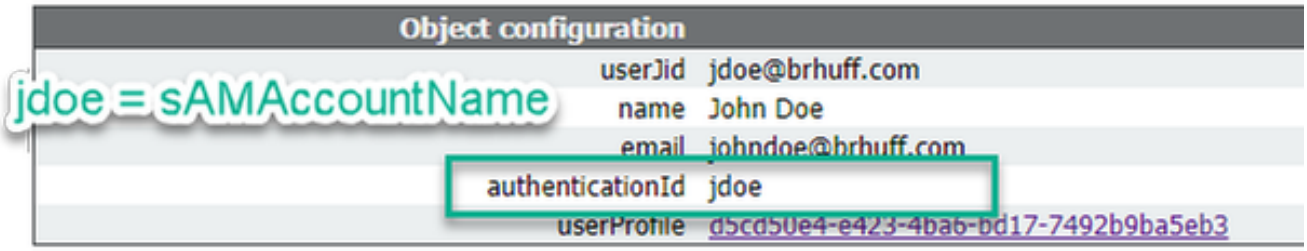
Note: The **authenticationIdMapping** is used by the callbridge/database to validate the claim sent from the IdP in the SAMLResponse and provide the user with a JSON Web Token (JWT).

4. Perform an **LDAP sync** on the **ldapSource** associated with the recently modified **ldapMapping**:

For Example:



5. After the LDAP sync is completed, navigate in the CMS API in **Configuration > api/v1/users** and select a user that was imported and verify the **authenticationId** is populated correctly.



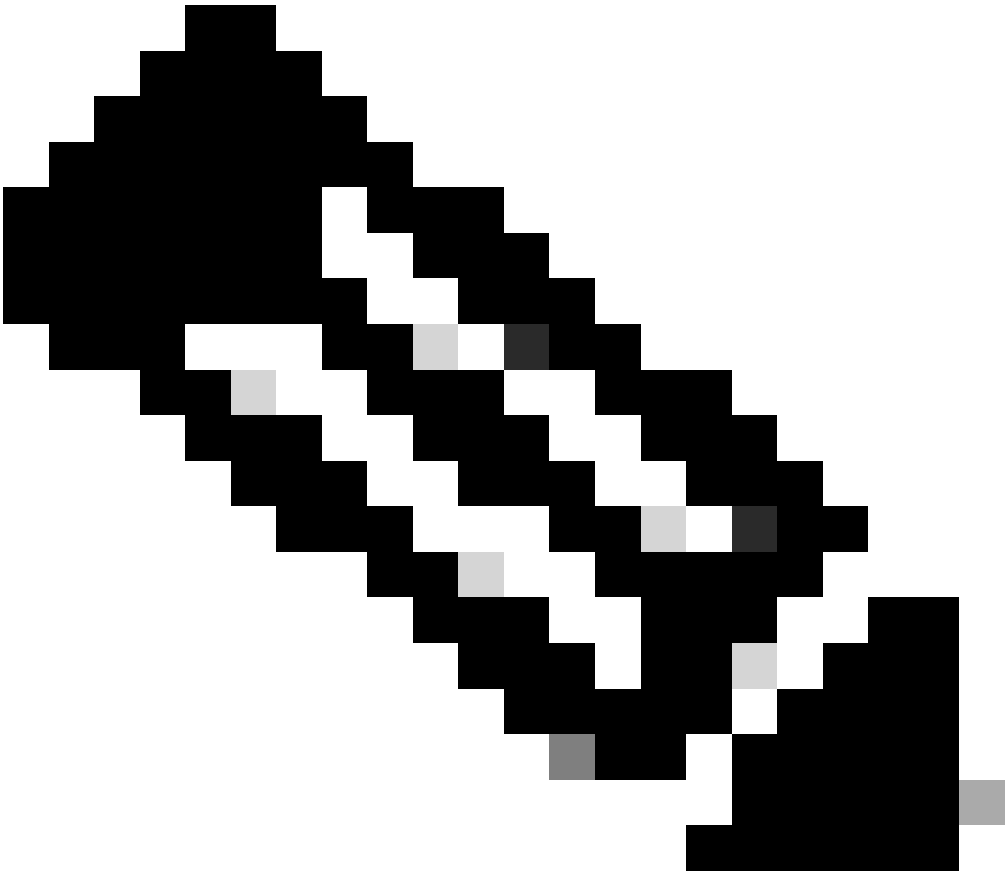
Create Webbridge Metadata XML for IdP

The Microsoft ADFS allows a Metadata XML file to be imported as a Relying Trust Party to identify the Service Provider being used. There are a few ways to create the Metadata XML file for this purpose, however, there are a few attributes that must be present in the file:

Example of Webbridge Metadata with required values:

```
<?xml version="1.0"?>
- <md:EntityDescriptor entityID="https://meet.brhuff.local:443" ID="https://meet.brhuff.local:443"
xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
  - <md:SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol" WantAssertionsSigned="true"
    AuthnRequestsSigned="false">
      <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</md:NameIDFormat>
      <md:AssertionConsumerService index="0" Location="https://meet.brhuff.local:443/api/auth/sso/idpResponse"
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"/>
    </md:SPSSODescriptor>
</md:EntityDescriptor>
```

1. **entityID** - This is the Webbridge3 server address (FQDN/Hostname) and associated port that is reachable by browsers for users.



Note: If there are multiple Webbridges using a single URL, this must be a load balancing address.

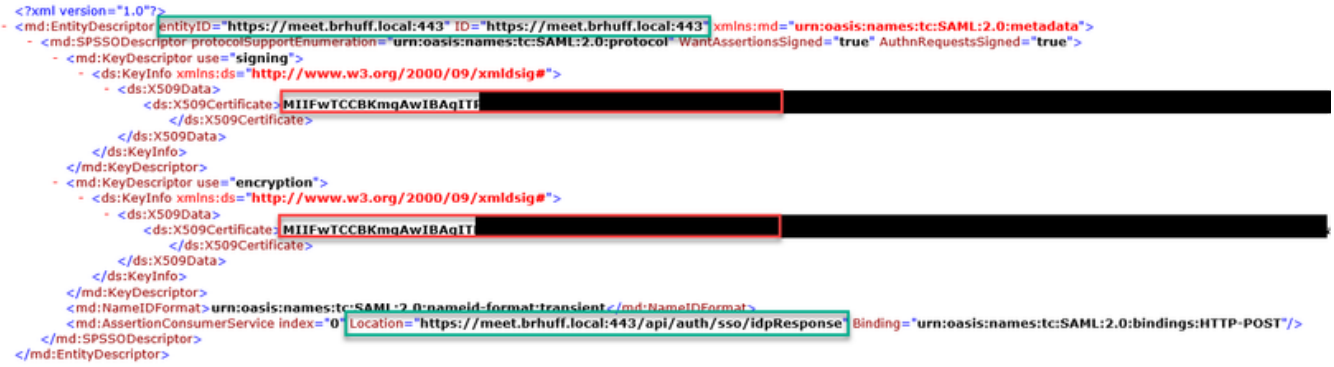
2. **Location** - This is the location in which the HTTP-POST AssertionConsumerService for the Webbridge Address. This is what tells the IdP where to redirect an authenticated user after sign-in. This must be set to the idpResponse URL: https://<WebbrdgeFQDN>:<port>/api/auth/sso/idpResponse. For example, https://join.example.com:443/api/auth/sso/idpResponse.

3. **OPTIONAL - Public Key for Signing** - this is the public key (certificate) for signing, which is be used by the IdP to verify AuthRequest from Webbridge. This MUST match with the private key 'sso_sign.key' on the SSO bundle uploaded on Webbridge so that the IdP can use the public key (certificate) to verify the signature. You can use an existing certificate from your deployment. Open the certificate in a text file and copy the content into the Webbridge Metadata file. Use the matching key for the certificate used in your sso_xxxx.zip file as the sso_sign.key file.

4. **OPTIONAL - Public Key for Encryption** - this is the public key (certificate) that the IdP is use to encrypt SAML information sent back to Webbridge. This MUST match the private key 'sso_encrypt.key' on the SSO bundle uploaded on Webbridge, so that Webbridge can decrypt what is sent back by IdP. You can use an existing certificate from your deployment. Open the certificate in a

text file and copy the content into the Webbridge Metadata file. Use the matching key for the certificate used in your sso_xxxx.zip file as the sso_encrypt.key file.

Example of Webbridge Metadata to be imported into IdP with optional public key (certificate) data:

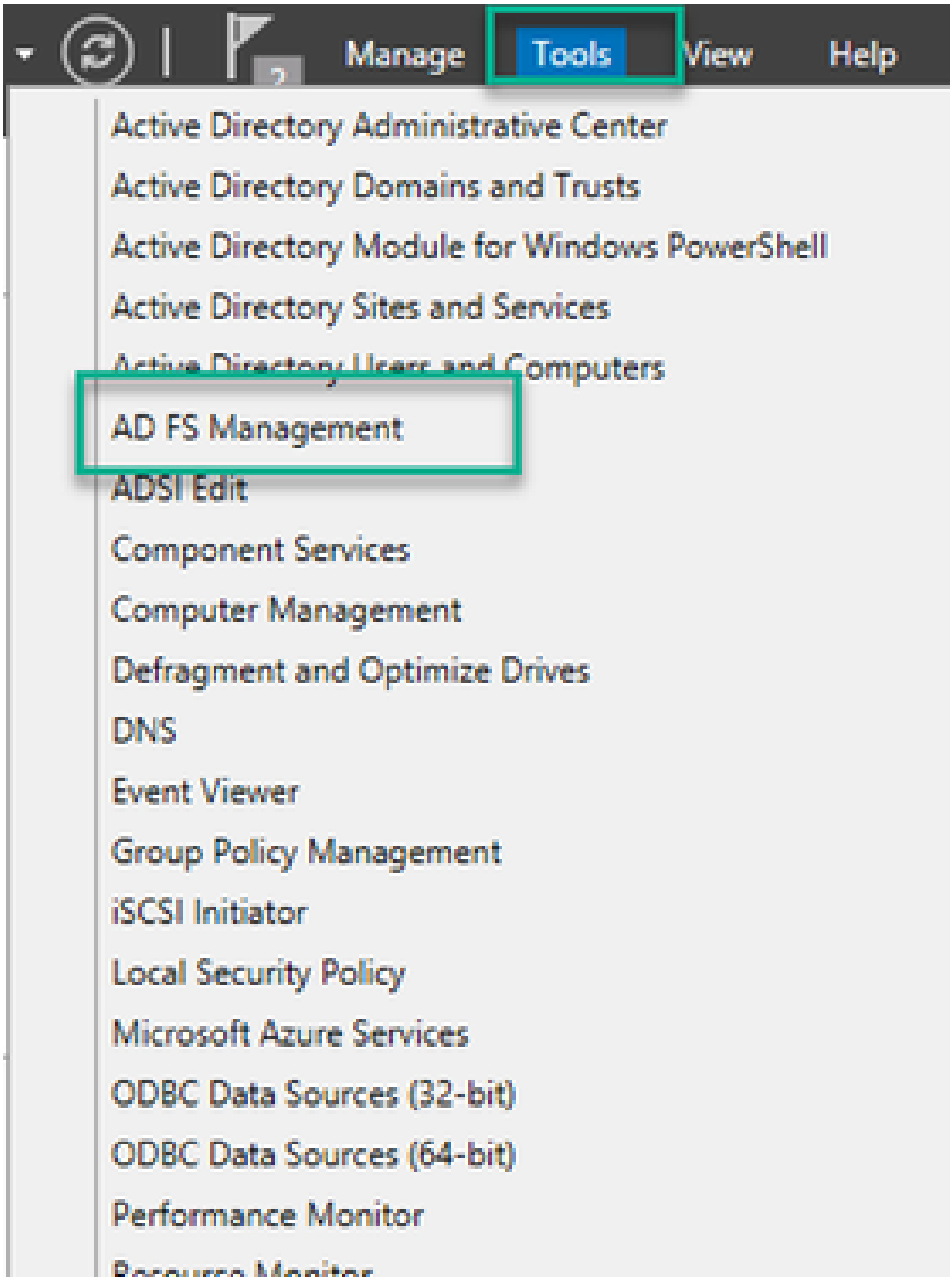


Import Metadata for Webbridge into Identity Provider (IdP)

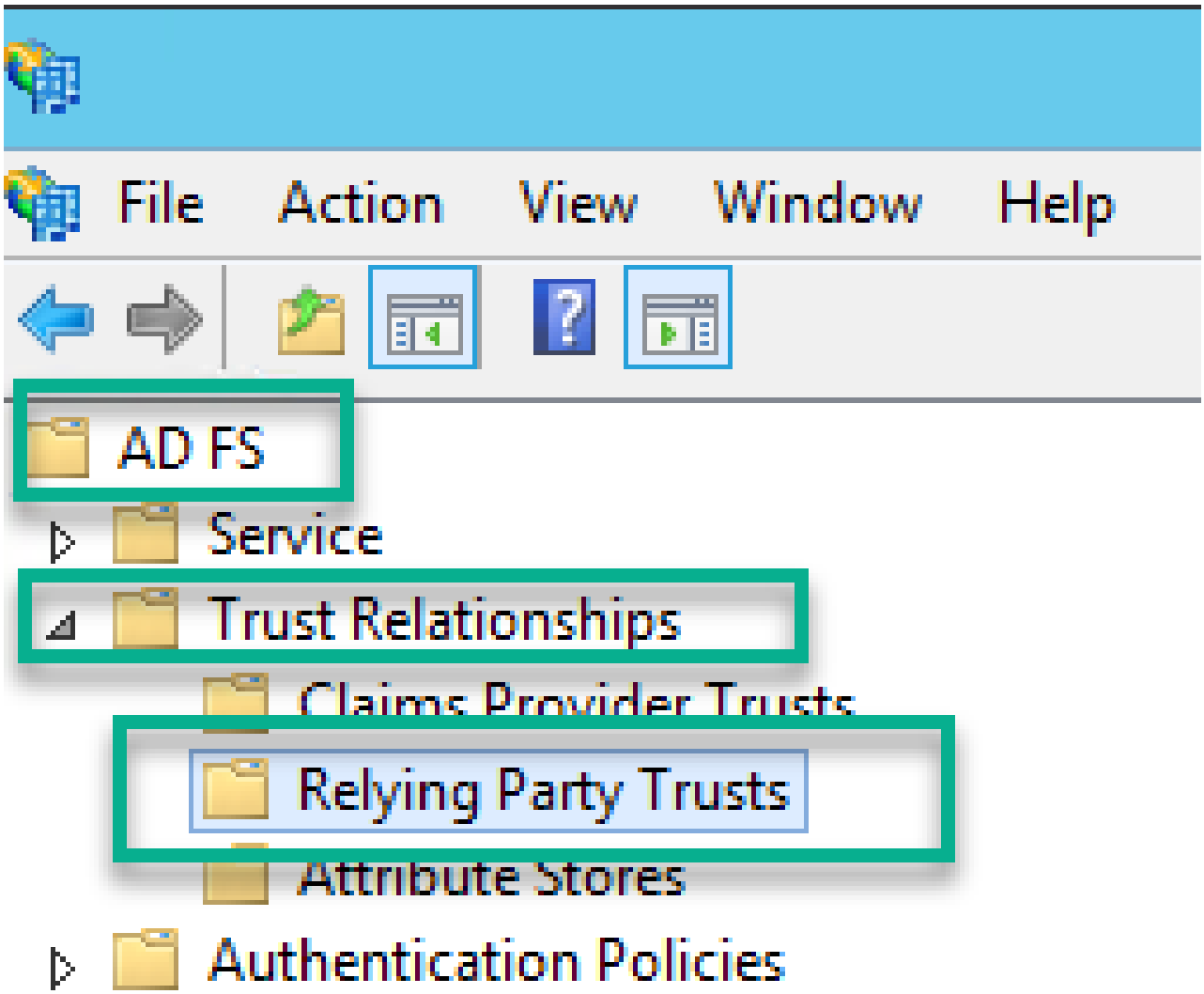
Once the Metadata XML has been created with the proper attributes the file can be imported into the Microsoft ADFS server to create a Relying Trust Party.

1. Remote Desktop into the Windows Server hosting the ADFS services

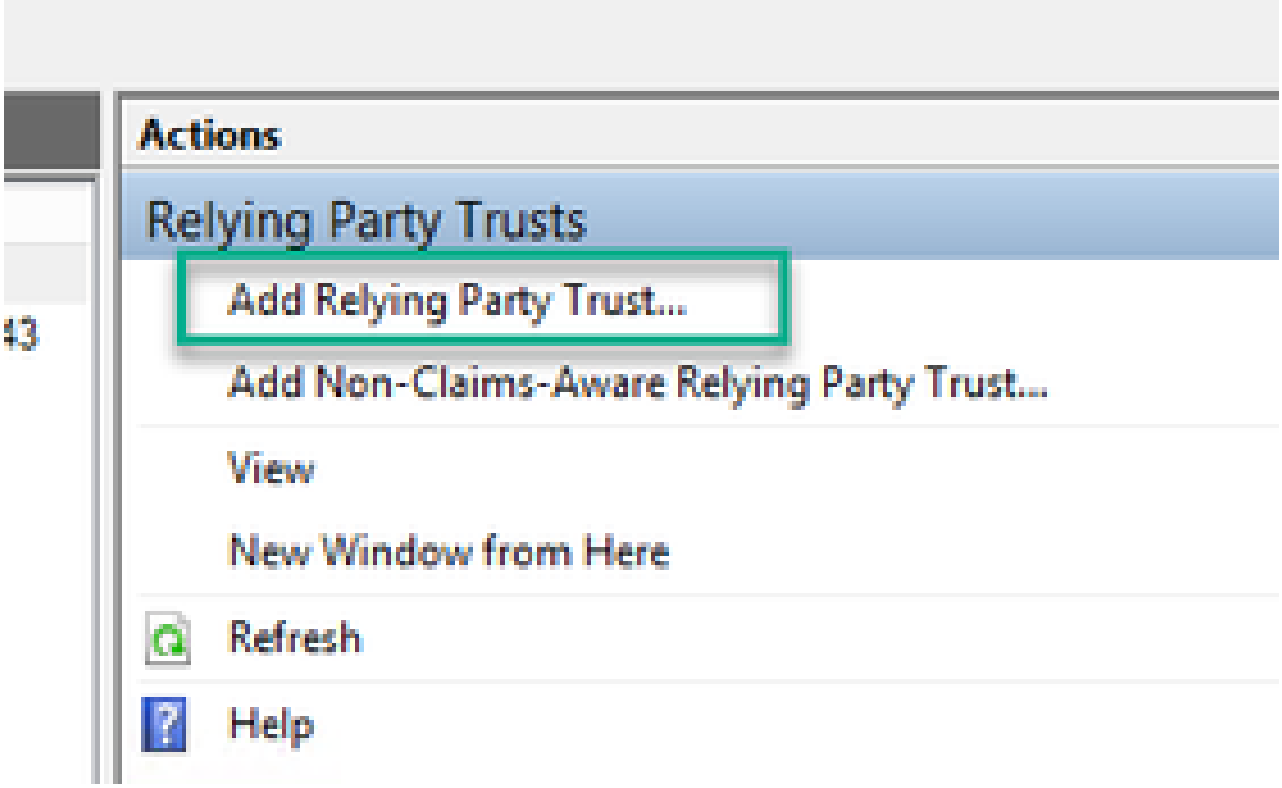
2. Open the AD FS Management Console, which can usually be accessed through the Server Manager.



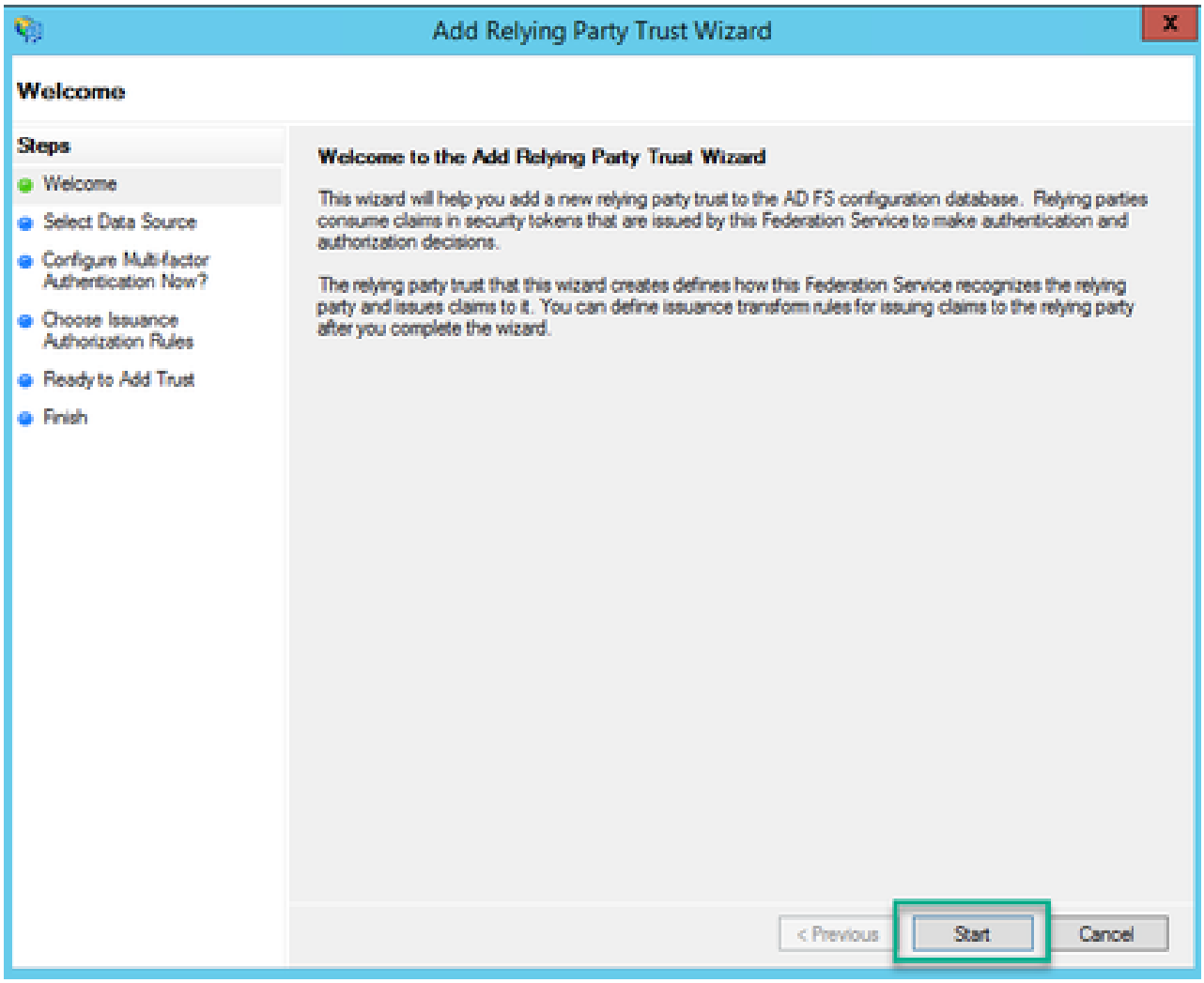
3. Once in the ADFS Management console, navigate to **ADFS > Trust Relationships > Relying Party Trust** in the Left pane.



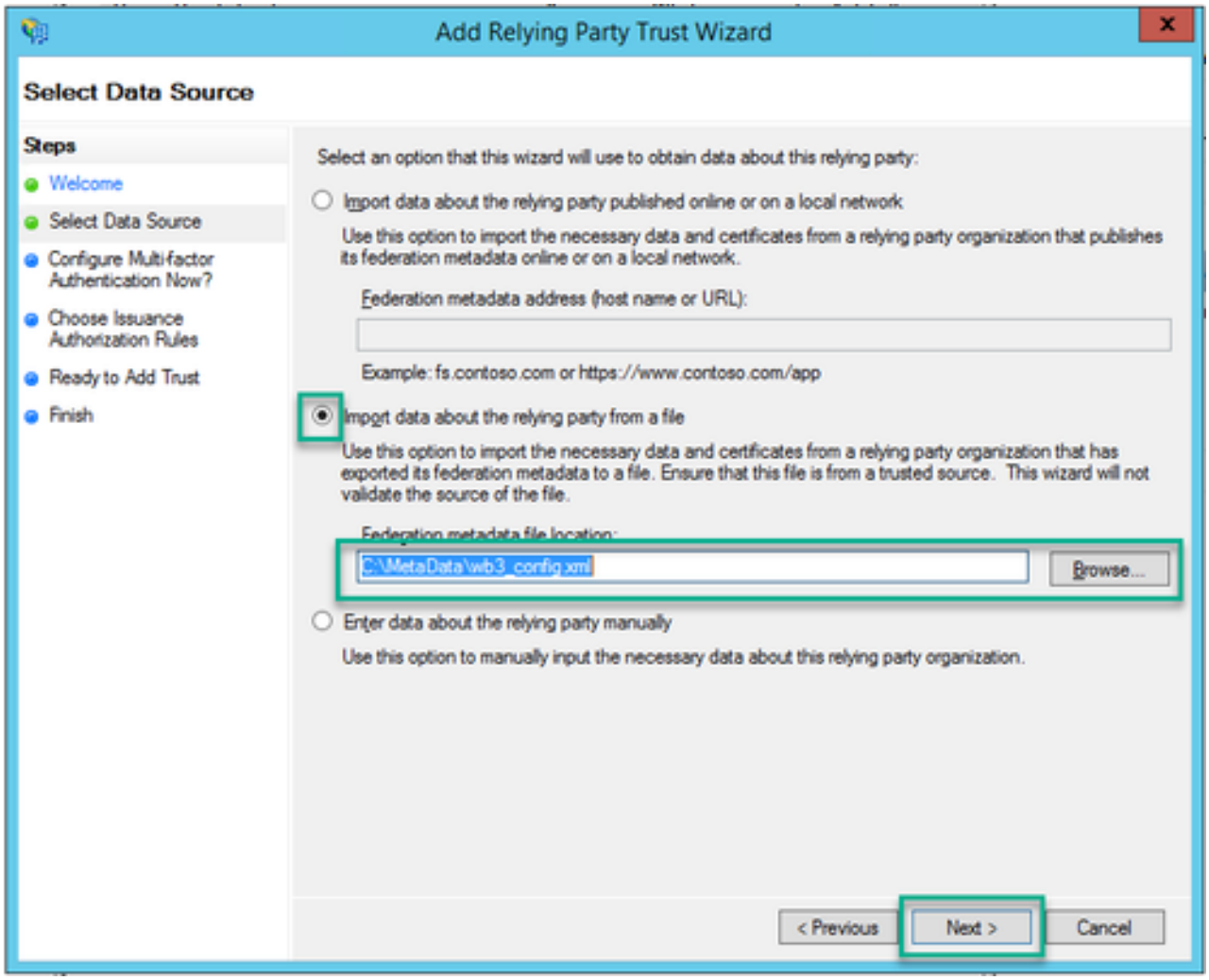
4. In the Right pane of the ADFS Management Console, select the **Add Relying Party Trust...** option.



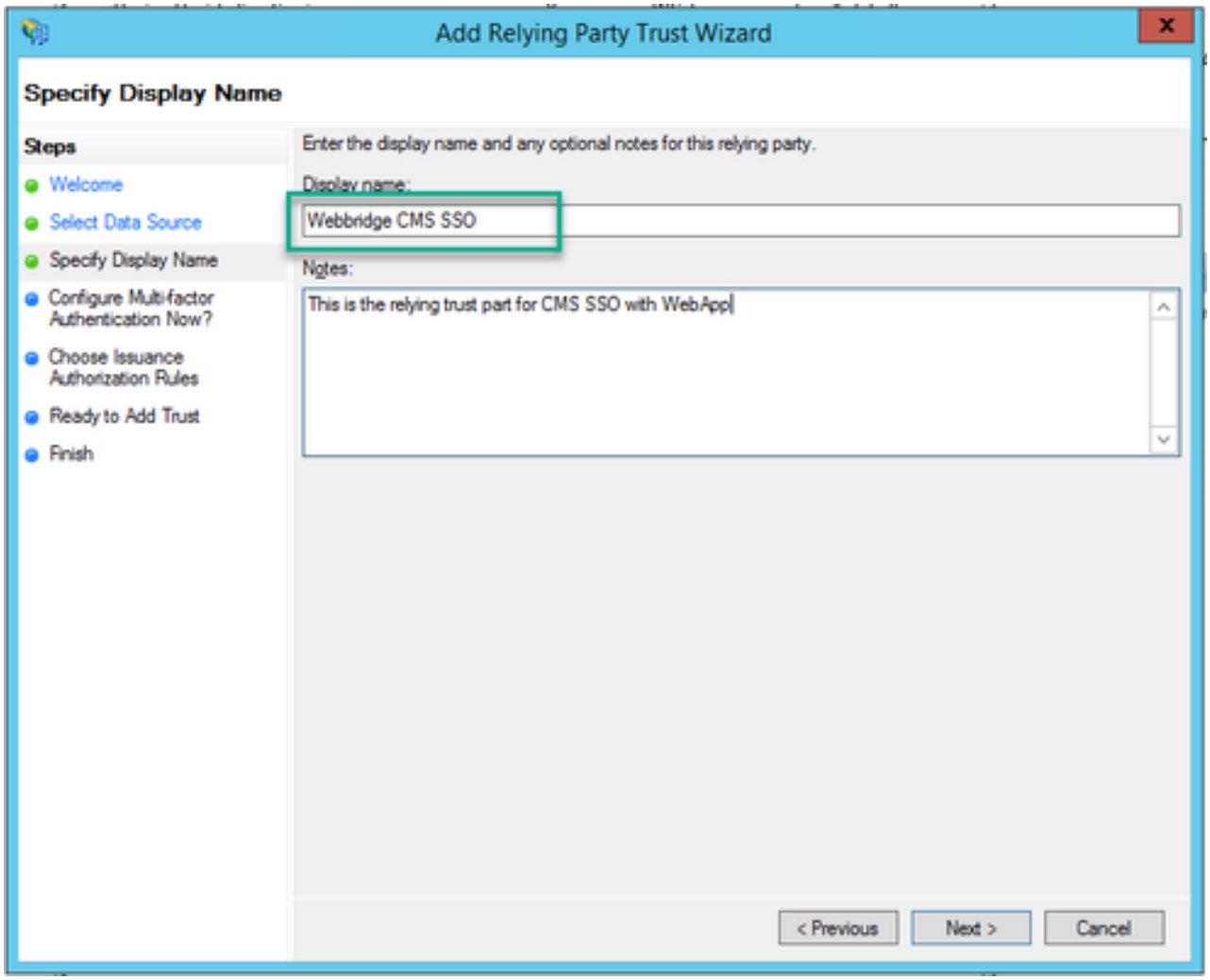
5. After making this select, the **Add Relying Party Trust Wizard** opens. Select the **Start** option.



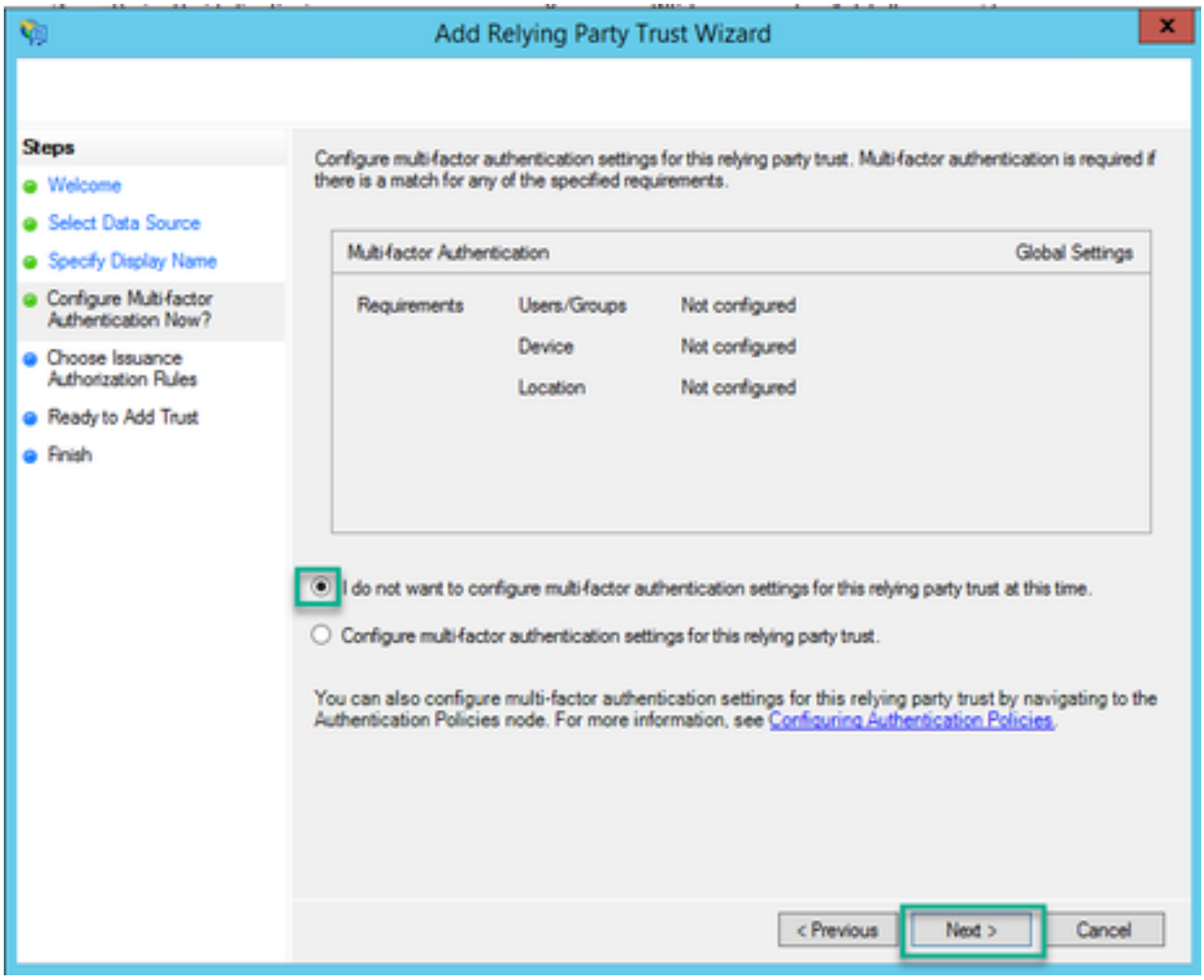
6. On the **Select Data Source** page, select the radio button for **Import data about the relying party from a file** and select **Browse** and navigate to the location of the Webbridge MetaData file.



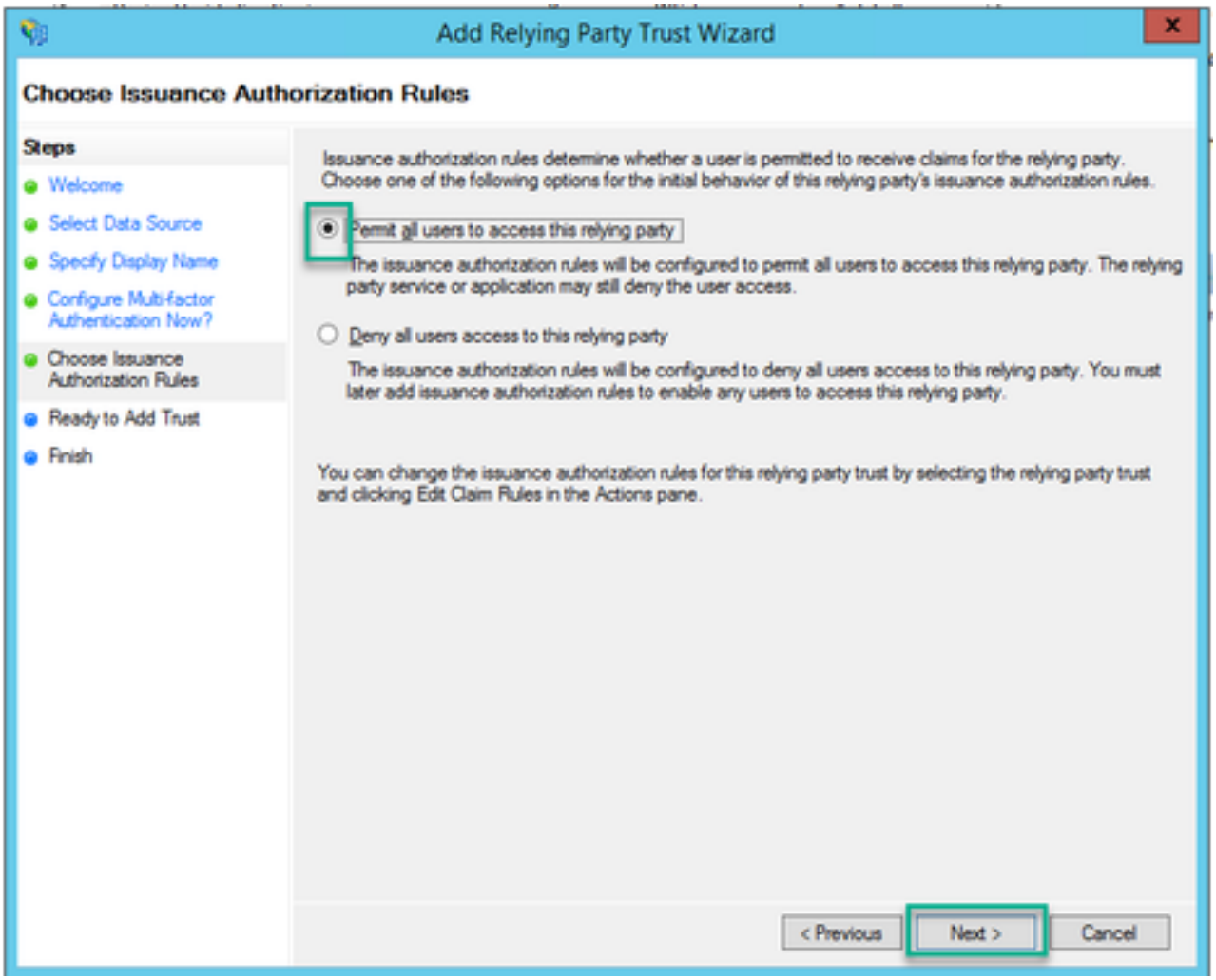
7. On the **Specify Display Name** page, put a name to be displayed for the entity in ADFS (the Display Name does not server purpose for the ADFS communication and is purely informational).



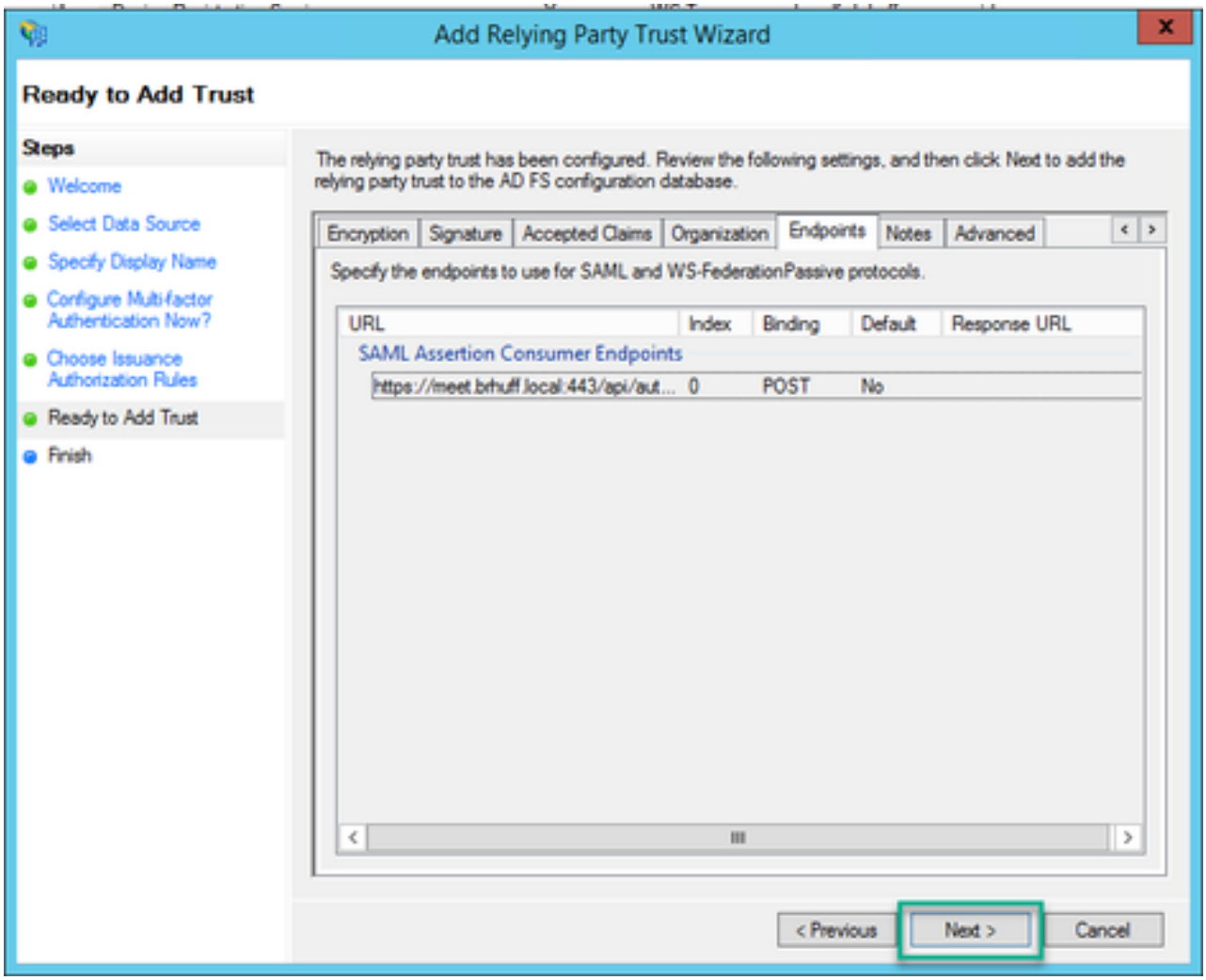
8. On the **Configure Multi-factor Authentication Now?** page, leave as default and select **Next**.



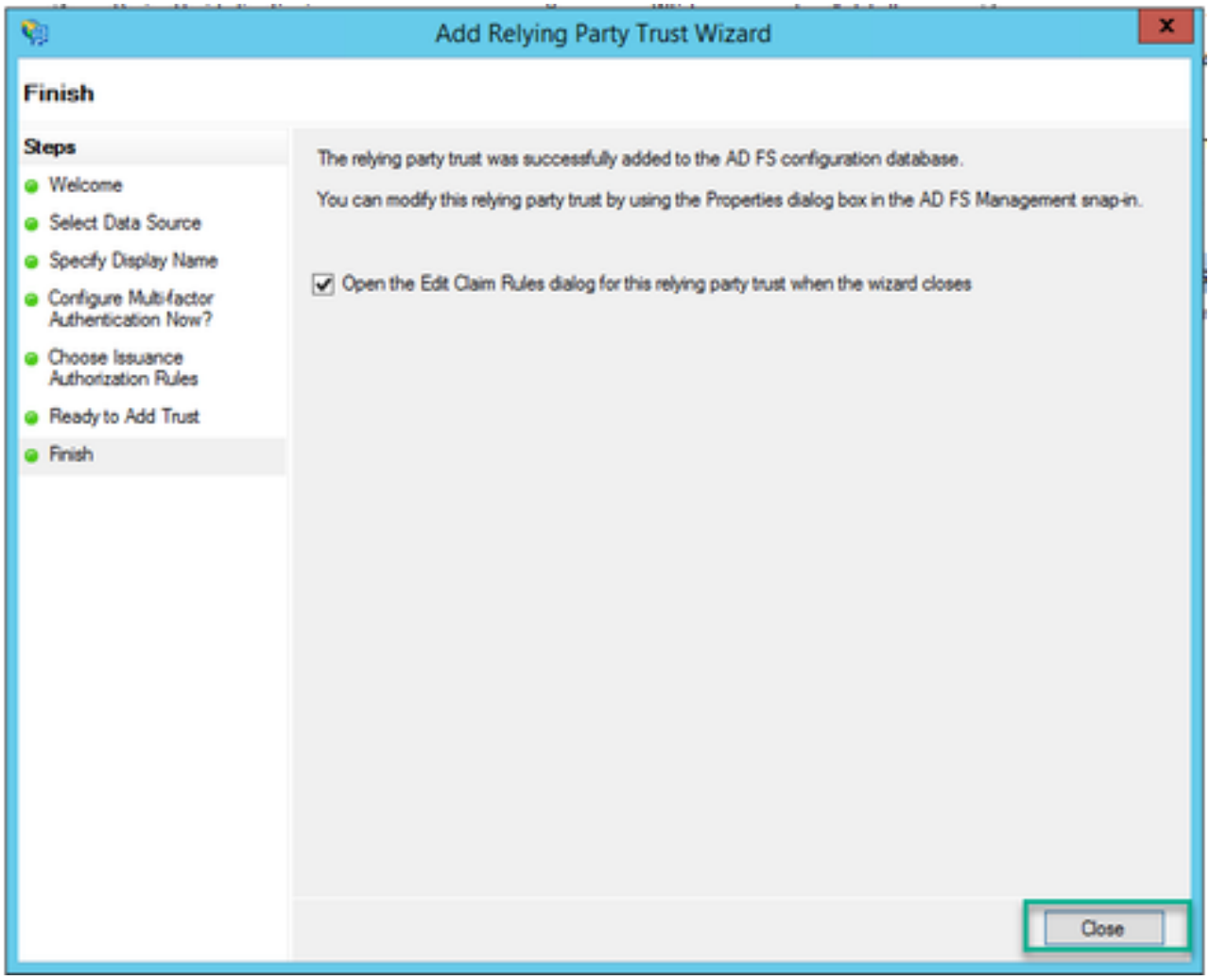
9. On the **Choose Issuance Authorization Rules** page, leave as selected for **Permit all users to access this relying party.**



10. On the **Ready to Add Trust** page, the imported details of the Relying Trust Party for Webbridge can be reviewed through the tabs. Check the **Identifiers** and **Endpoints** for the URL details for Webbridge Service Provider.



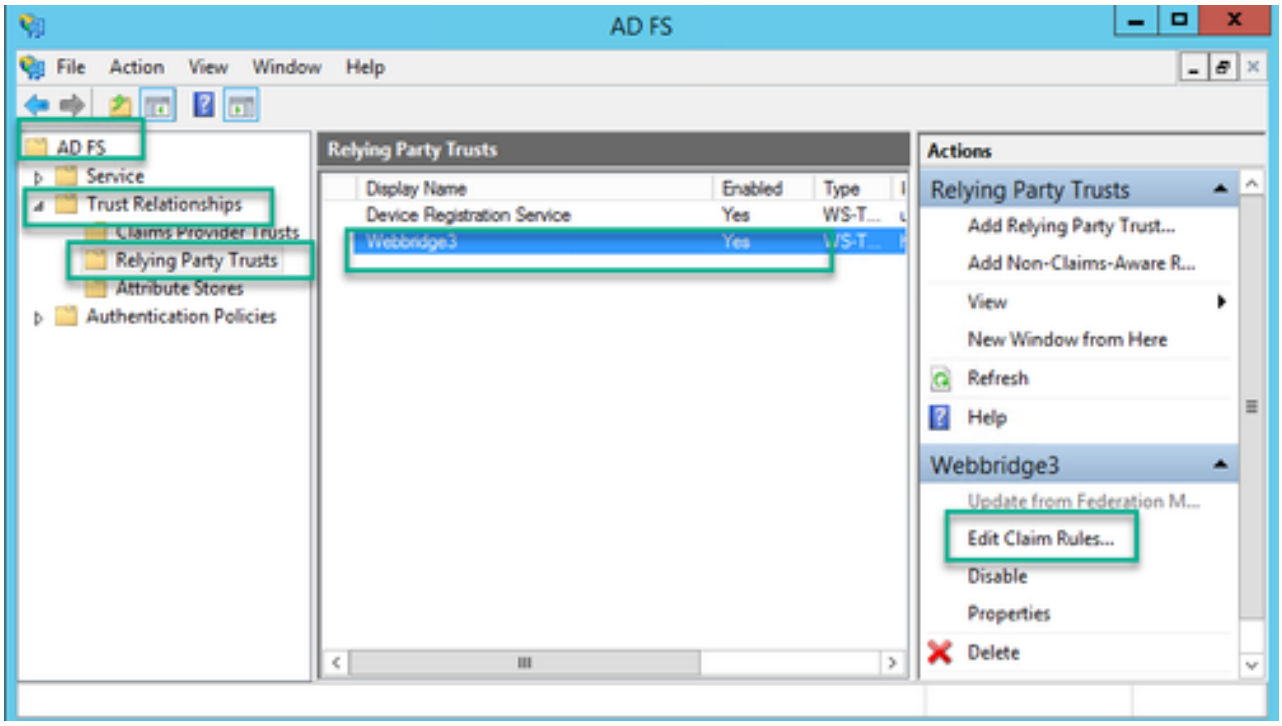
11. On the **Finish** page, select **Close** option to close the wizard and continue with editing claim rules.



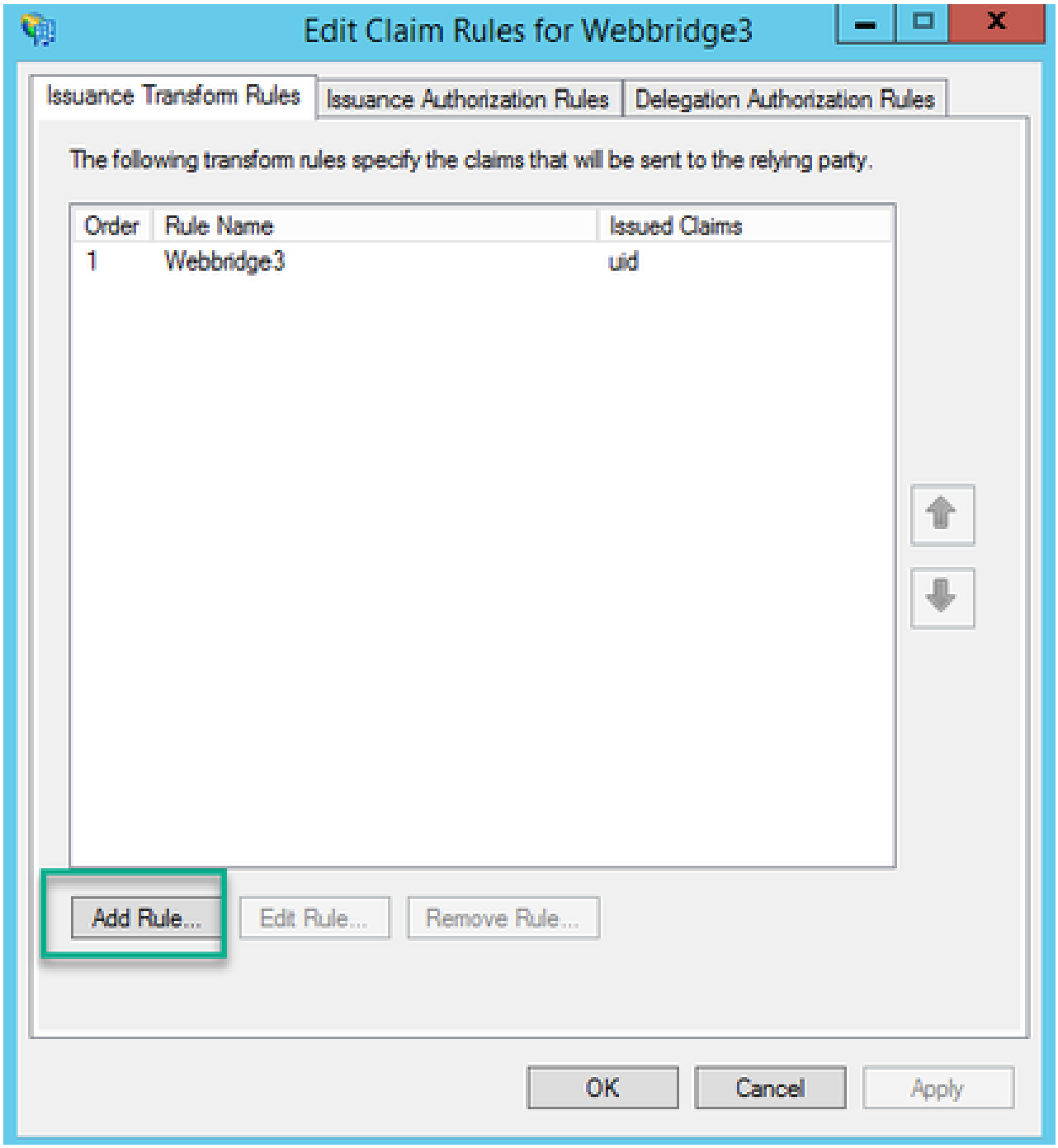
Create Claim Rules for the Webbridge Service on the IdP

Now that the Relying Party Trust has been created for the Webbridge, claim rules can be created to match specific LDAP Attributes to outgoing claim types to be provided to the Webbridge in the SAML Response.

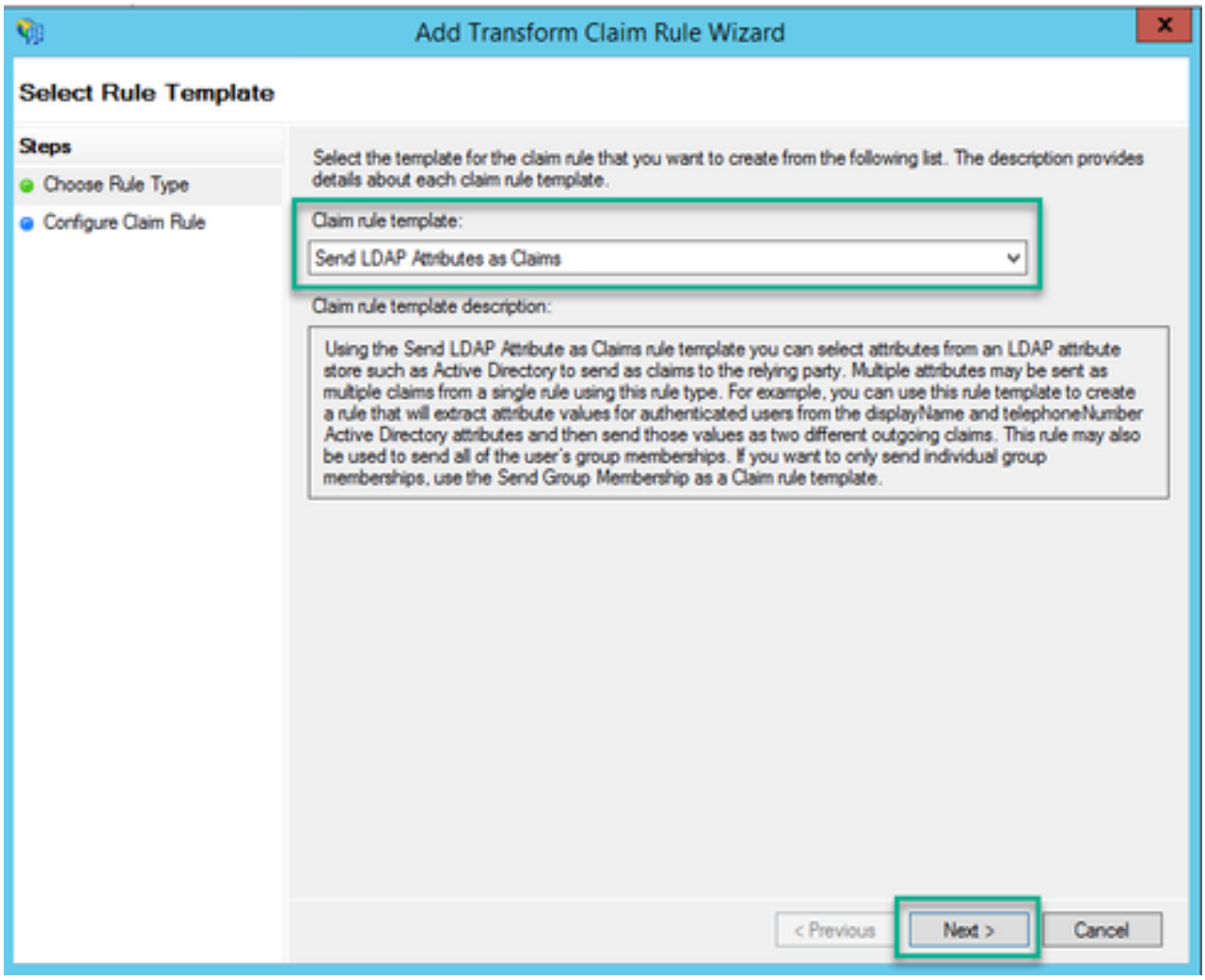
1. In the **ADFS Management console**, highlight the **Relying Party Trust** for the Webbridge and select **Edit Claim Rules** on the right pane.



2. On the **Edit Claim Rules for <DisplayName>** page, select the **Add Rule...**.

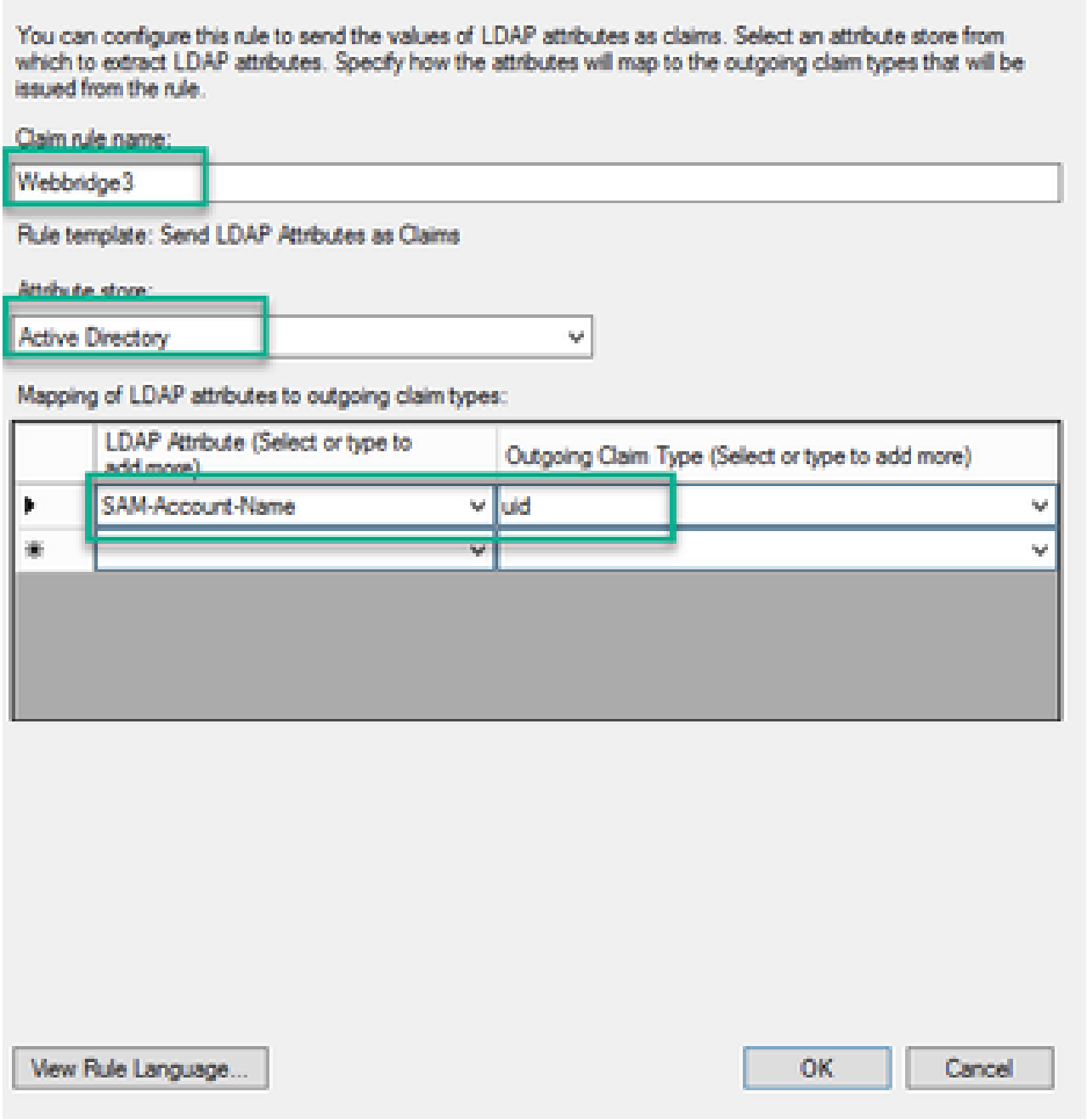


3. On the **Add Transform Claim Rule Wizard** page, select **Send LDAP Attributes as Claims** for the Claim rule template option and select **Next**.



4. On the **Configure Claim Rule** page, configure the claim rule for the **Relying Party Trust** with these values:

1. **Claim rule name** = this must be a name given to the rule in ADFS (for rule reference only)
2. **Attribute store** = Active Directory
3. **LDAP Attribute** = This must match the **authenticationIdMapping** in the Callbridge API. (For example, $sAMAccountName$.)
4. **Outgoing Claim Type** = This must match the **authenticationIdMapping** in the Webbridge SSO **config.json**. (For example, uid.)



Create SSO Archive ZIP file for Webbridge:

This configuration is what the Webbridge references to validate the SSO configuration for supported domains, authentication mapping, and so on. These rules must be considered for this part of the configuration:

- The ZIP file MUST start with **sso_** prefixed to the file name (for example, **sso_cmstest.zip**).
- Once this file is uploaded, Webbridge disables basic authentication and ONLY SSO can be used for the Webbridge this has been uploaded to.
- If there are multiple Identity Providers used, a separate ZIP file must be uploaded with a different naming schema (STILL prefixed with the **sso_**).
- When creating the zip file, be sure to highlight and zip the file contents and do not put the required files into a folder and zip that folder.

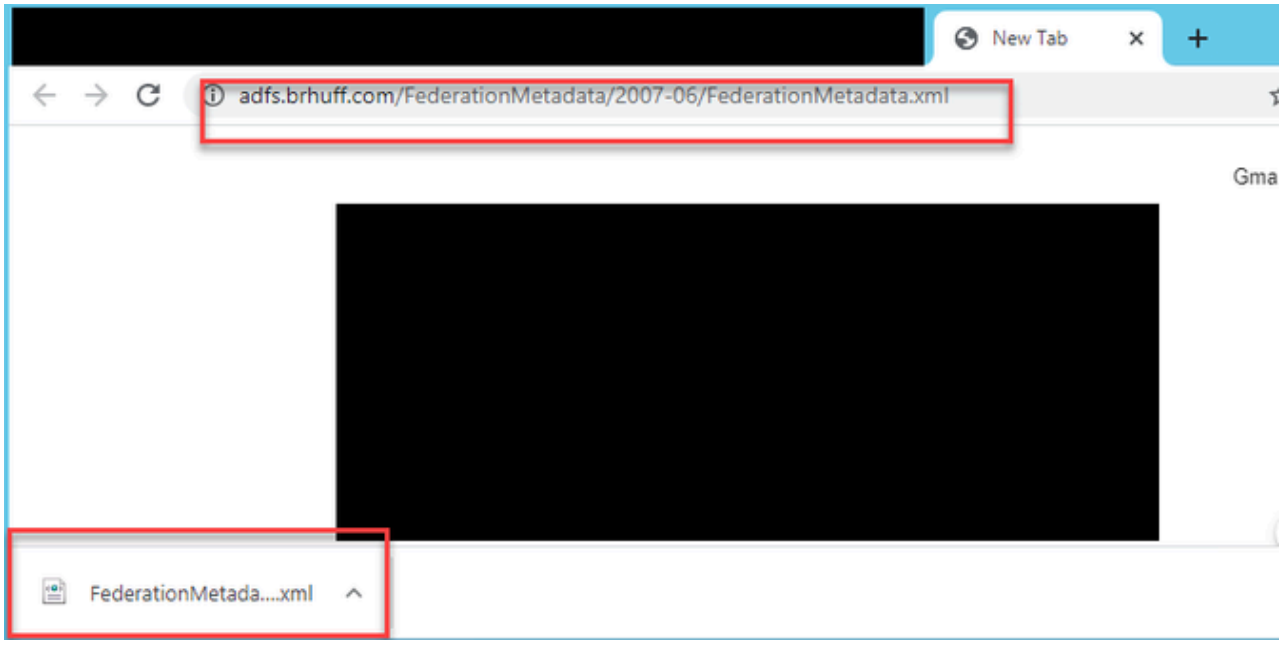
The contents of the zip file are made up of 2 to 4 files, depending if encryption is being used or not.

Filename	Description	Required?
idp_config.xml	This is the MetaData file that can be collected by the idP. In ADFS this can be located by going to https://<ADFSFQDN>/FederationMetadata/2007-06/FederationMetadata.xml.	YES
config.json	This is the JSON file in which Webbridge uses to validate the supported domains, authentication mapping for SSO.	YES
sso_sign.key	This is the private key for public signing key configured on the Identify Provider. Only needed for securing the signed data	NO
sso_encrypt.key	This is the private key for public encrypting key configured on the Identify Provider. Only needed for securing the encrypted data	NO

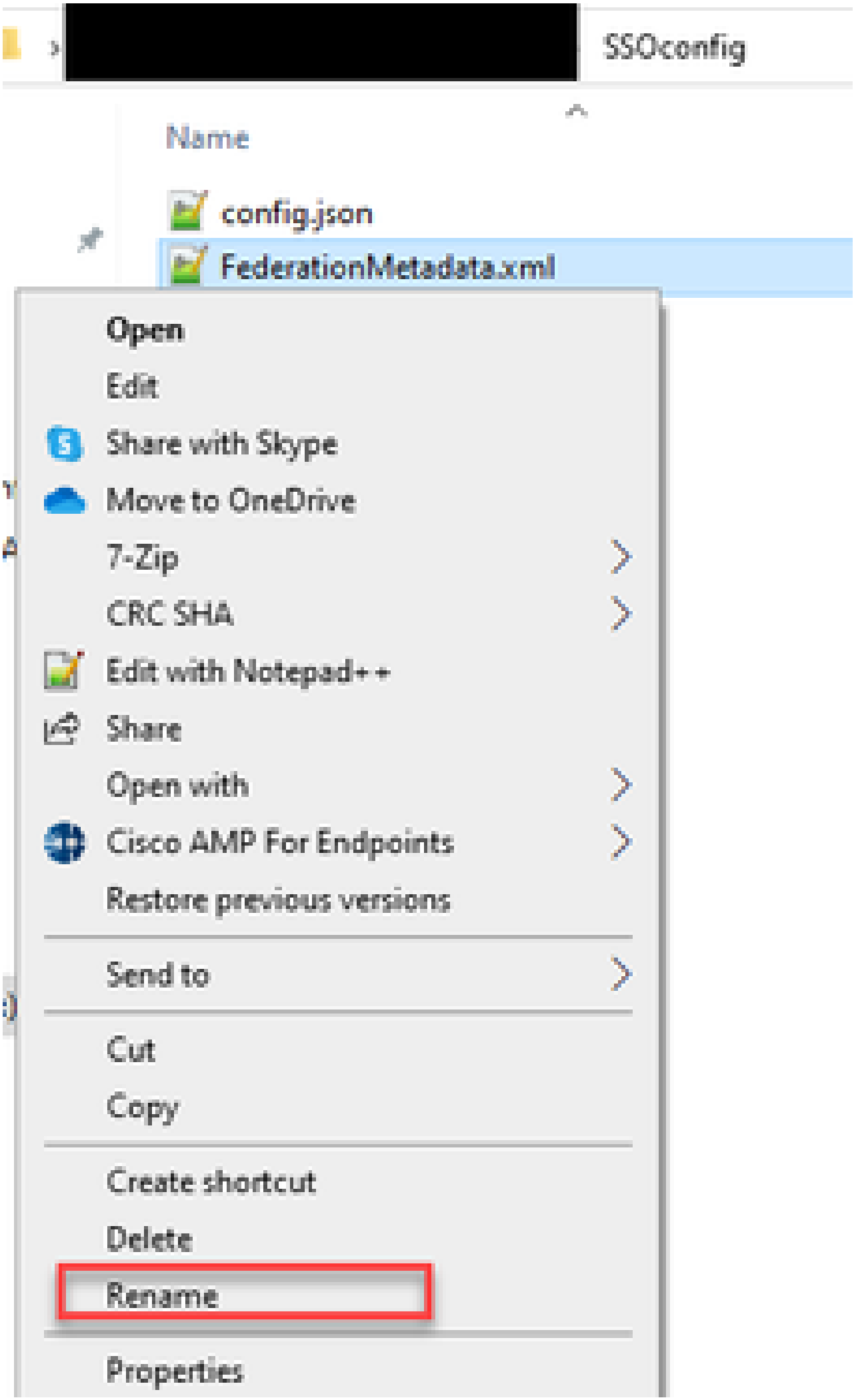
Obtain and configure the idp_config.xml

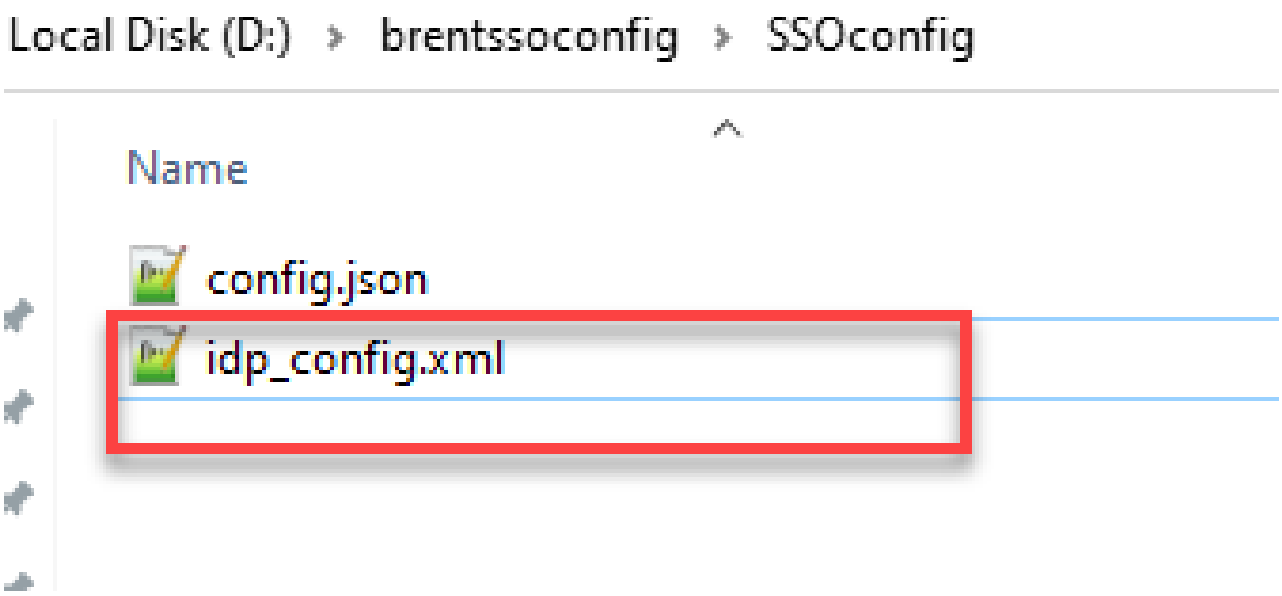
1. On the ADFS server (or a location that has access to the ADFS), open a Web Browser.

2. In the Web Browser, enter URL: https://<ADFSFQDN>/FederationMetadata/2007-06/FederationMetadata.xml (You can also use localhost instead of the FQDN if you are locally on the ADFS server). This downloads the file **FederationMetadata.xml**.



3. Copy downloaded file to a location where the zip file is being created and rename to **idp_config.xml**.

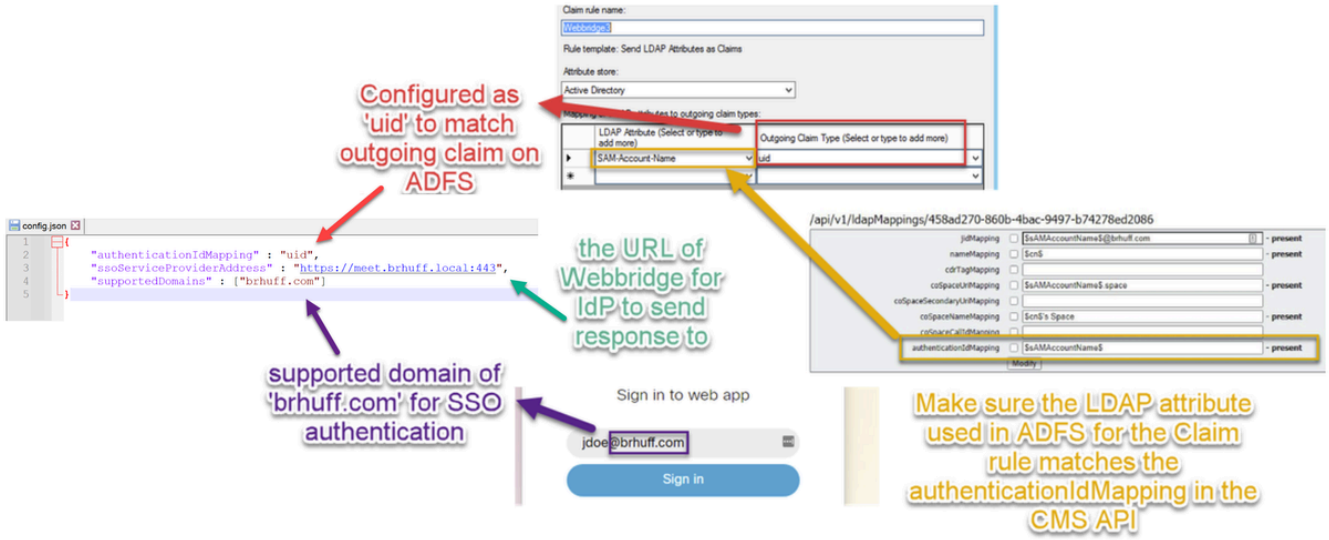

[redacted]
SSOconfig
Name
config.json
FederationMetadata.xml
Open
Edit
Share with Skype
Move to OneDrive
7-Zip
CRC SHA
Edit with Notepad++
Share
Open with
Cisco AMP For Endpoints
Restore previous versions
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties



Create the config.json File with Contents

The **config.json** contains the these 3 attributes and they must be contained within brackets, { }:

1. **supportedDomains** - This is a list of domains that are checked for SSO authentication against the IdP. Multiple domains can be separated by a comma.
2. **authenticationIdMapping** - This is the parameter that is passed back as a part of the outgoing claim rule from the ADFS/IdP. This must match the name value of the outgoing claim type on the IdP. Claim Rule.
3. **ssoServiceProviderAddress** - This is the FQDN URL to which the Identify Provider sends the SAML responses to. This must be the Webbridge FQDN.

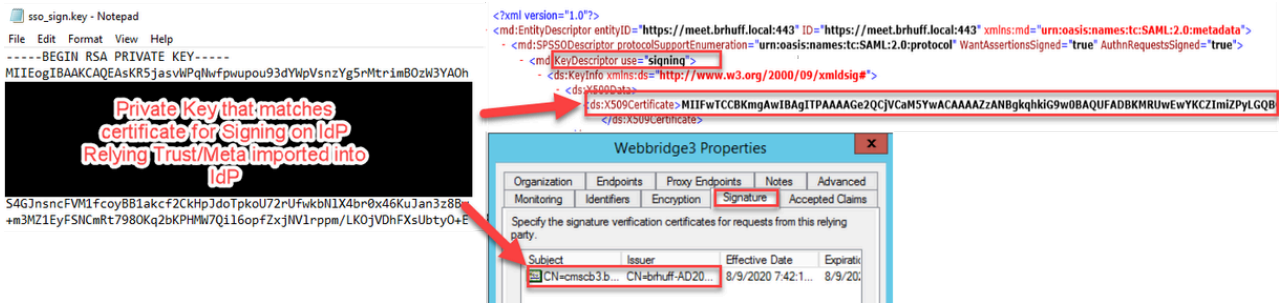


Set the sso_sign.key (OPTIONAL)

This file must contain the private key of the certificate used for signing in the Webbridge metadata that was imported to the IdP. The certificate used for signing can be set during import of the Webbridge metadata in the ADFS by populating the **X509Certificate** with the certificate information under the **<KeyDescriptor use=signing>** section. It can also be viewed (and imported) on ADFS in the **Webbridge Relying Trust Party** under **Properties > Signature**.

In the next example, you can see the callbridge certificate (CN=**cmscb3.brhuff.local**), which was added to the Webbridge metadata prior to being imported into ADFS. The private key inserted into the **sso_sign.key** is the one that matches the **cmscb3.brhuff.local** certificate.

This is an optional configuration and only needed if intending to encrypt the SAML Responses.

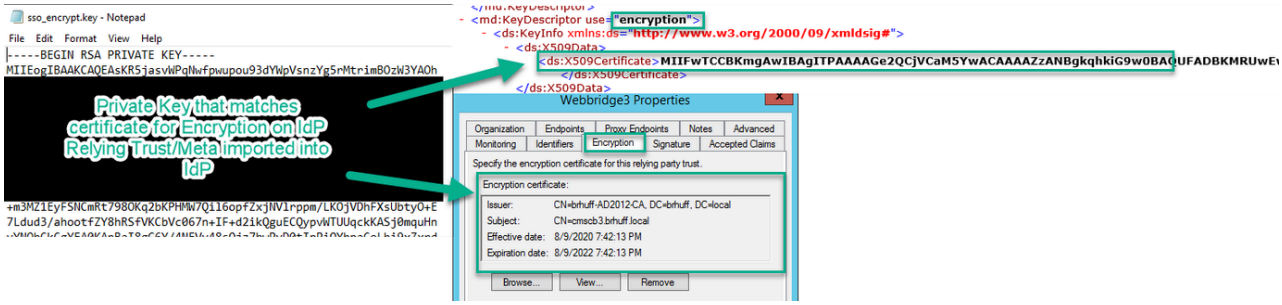


Set the sso_encrypt.key (OPTIONAL)

This file must contain the private key of the certificate used for encryption in the webbridge metadata that was imported to the IdP. The certificate used for encryption can be set during import of the Webbridge metadata in the ADFS by populating the **X509Certificate** with the certificate information under the **<KeyDescriptor use=encryption>** section. It can also be viewed (and imported) on ADFS in the **Webbridge Relying Trust Party** under **Properties > Encryption**.

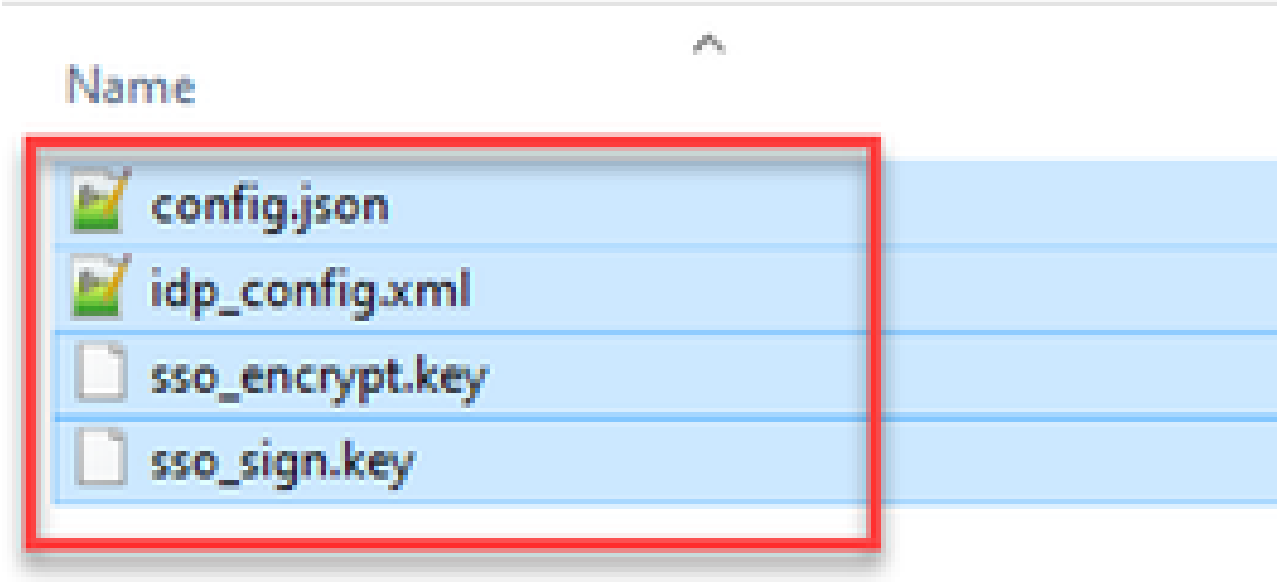
In the next example, you can see the callbridge certificate (CN=cmscb3.brhuff.local), which was added to the Webbridge metadata prior to being imported into ADFS. The private key inserted into the 'sso_encrypt.key' is the one that matches the **cmscb3.brhuff.local** certificate.

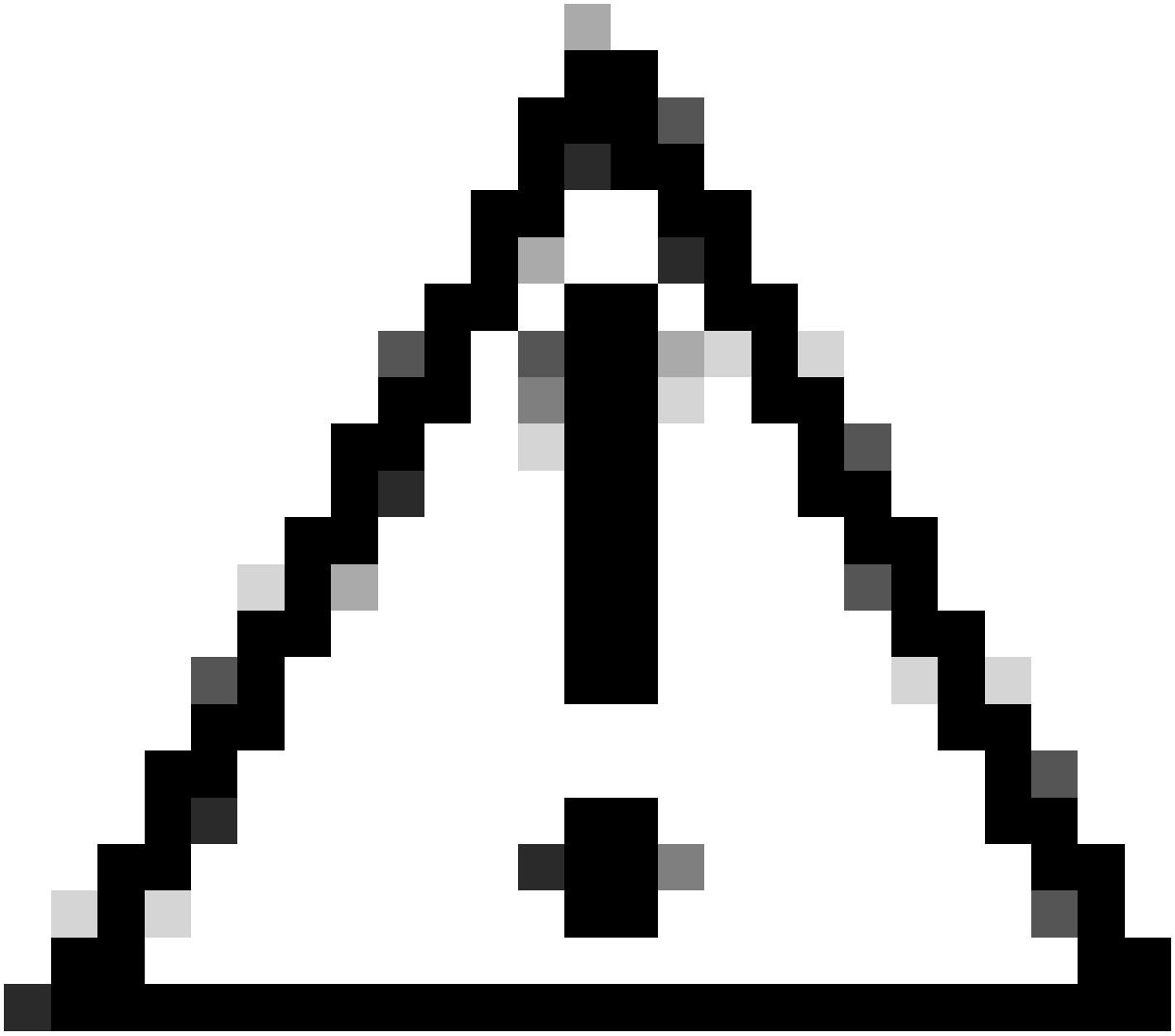
This is an optional configuration and it is only needed if you intend to encrypt the SAML Responses.



Creating the SSO ZIP file

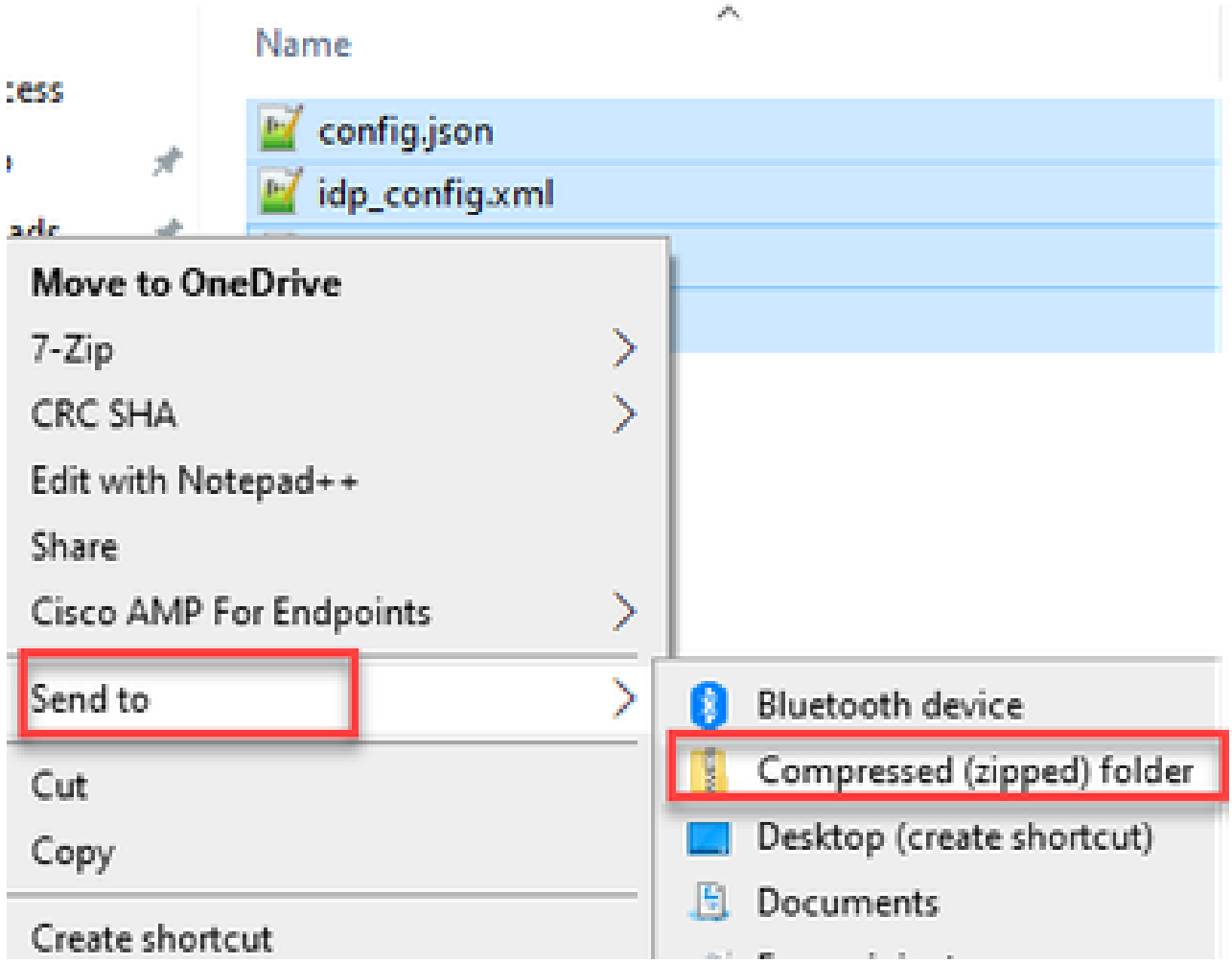
1. Highlight all the files intended to be used for the SSO config file.


Name
config.json
idp_config.xml
sso_encrypt.key
sso_sign.key

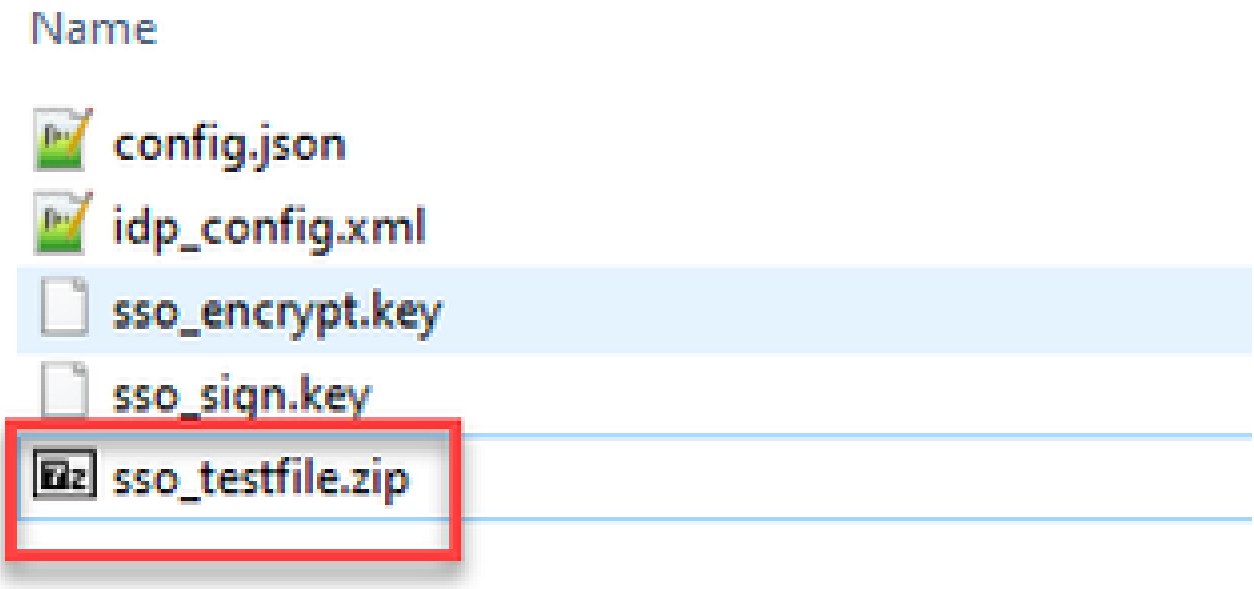


Caution: Do not zip the folder containing the files because this results in the SSO not working.

2. Right click on the highlight files and select **Send to > Compressed (zipped) folder.**



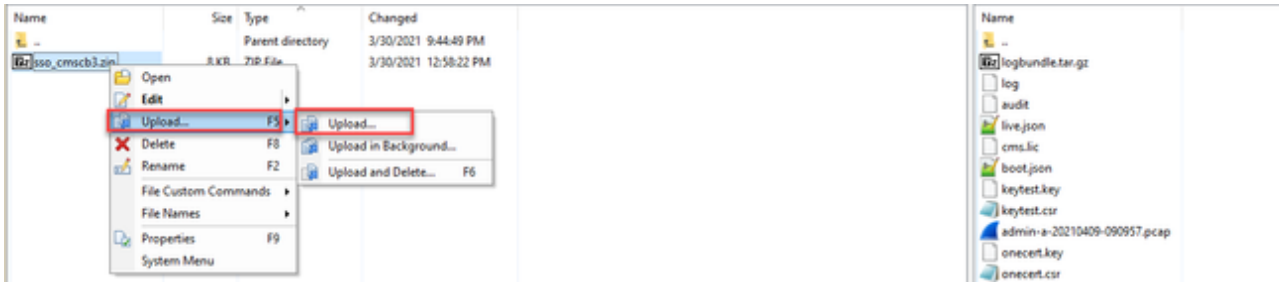
3. After the files have been zipped, rename them to the desired name with the **sso_** prefix:



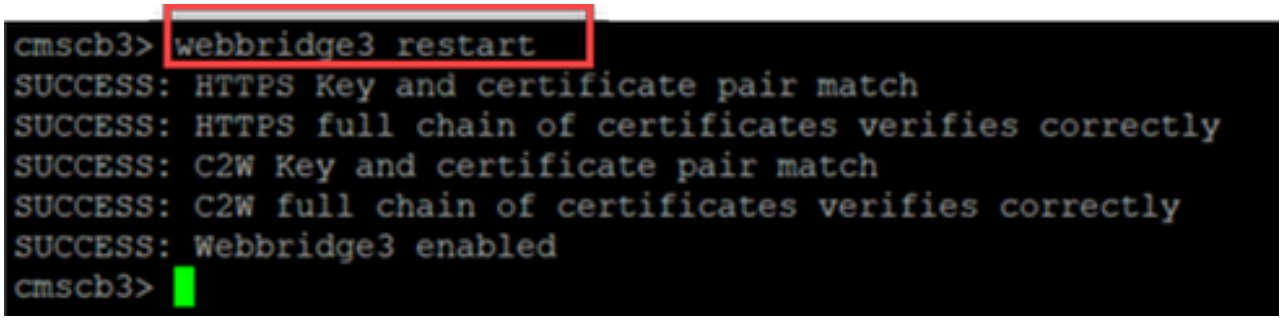
Upload the SSO Zip file(s) to Webbridge

Open an SFTP/SCP client, in this example WinSCP is being used, and connect to the server hosting Webbridge3.

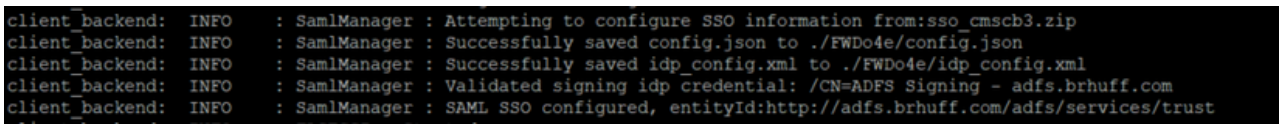
1. In the left pane, navigate to the location i n which the SSO Zip file resides and either right click select upload or drag and drop the file.



2. Once the file has been uploaded completely to the Webbridge3 server, open an SSH session and run the command **webbridge3 restart**.



3. In the syslog, these messages indicate the SSO enable was successful:

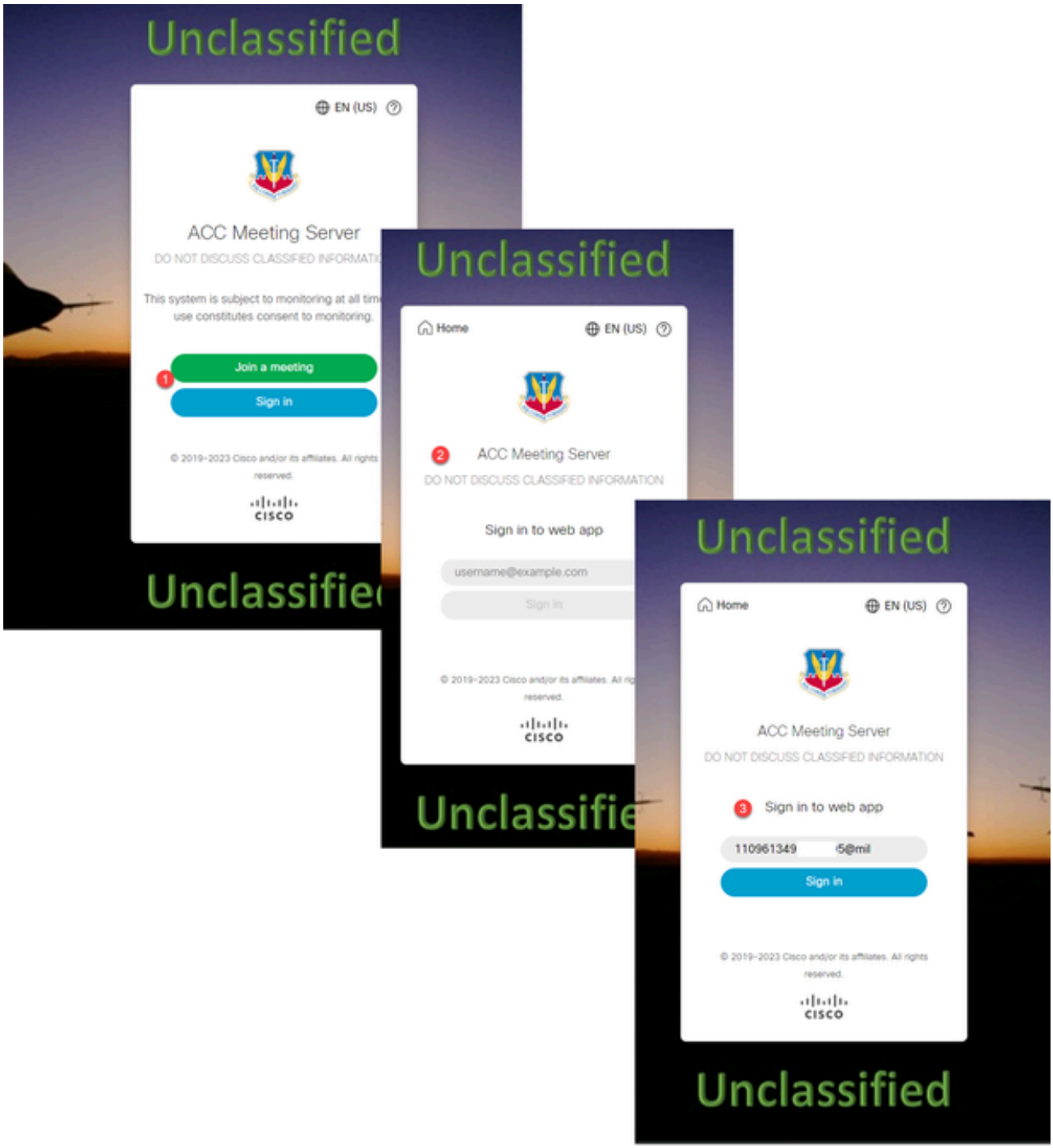


Common Access Card (CAC)

A Common Access Card (CAC) is a smart card that serves as the standard identification for active duty military personnel, Department of Defense civilian employees, and eligible contractor personnel.

Here is the entire sign-in process for users who use CAC Cards:

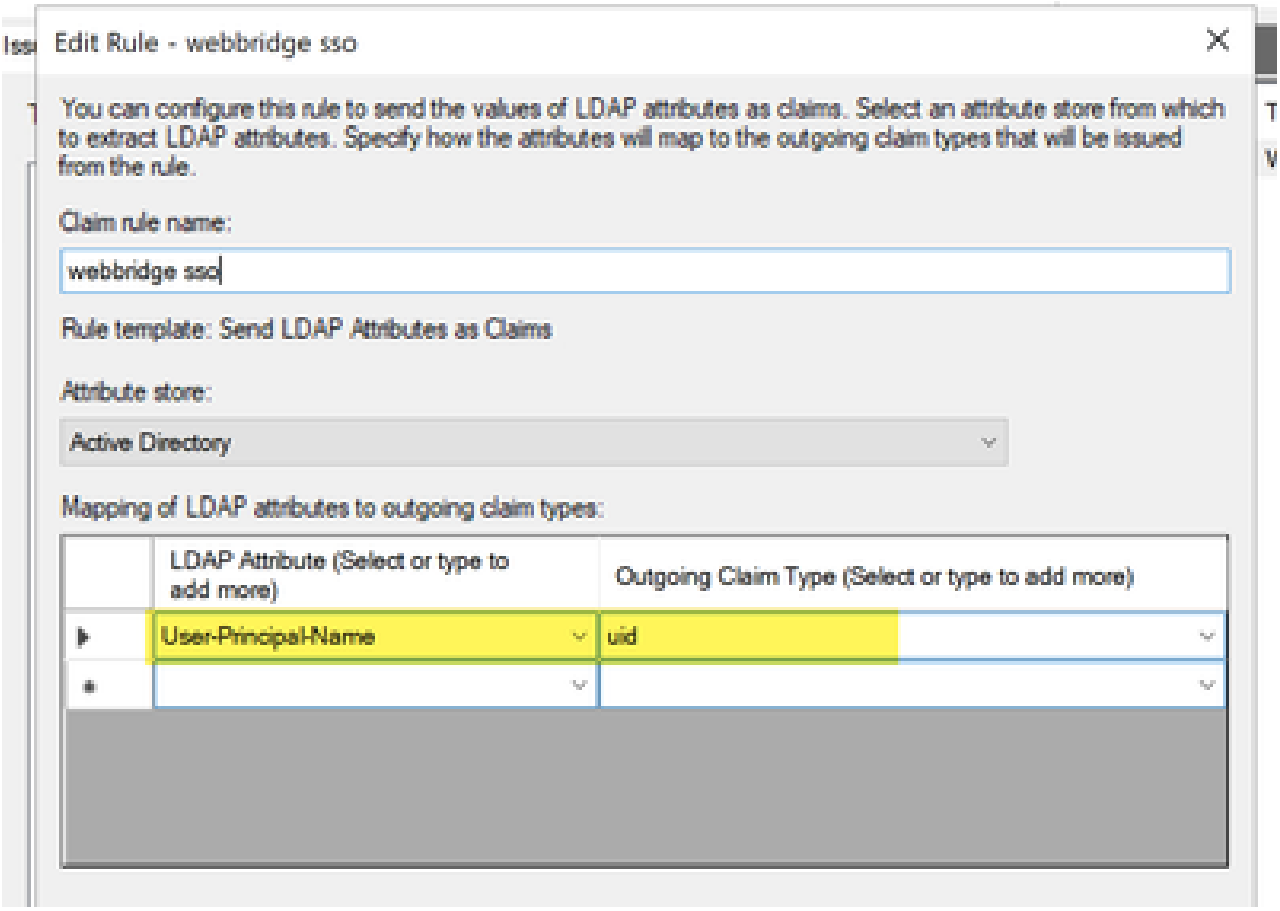
1. Turn on PC, and stick in CAC card
2. Log in (pick cert sometimes), and enter Pin
3. Open browser
4. Navigate to the join URL and see the **Join a meeting** or **Sign In** options
5. Sign in: Enter the username that is configured as jidMapping that Active Directory is expecting from a CAC login
6. Hit sign in
7. ADFS page comes up briefly and is auto populated
8. User are logged in at this point



Configure **jidMapping** (this is the users sign in name) in **Ldapmapping** the same as what ADFS uses for CAC card. **$userPrincipalName$** for example (case sensitive)

Also set the same LDAP attribute for the **authenticationIdMapping** to match the attribute which is used in the Claim rule in ADFS.

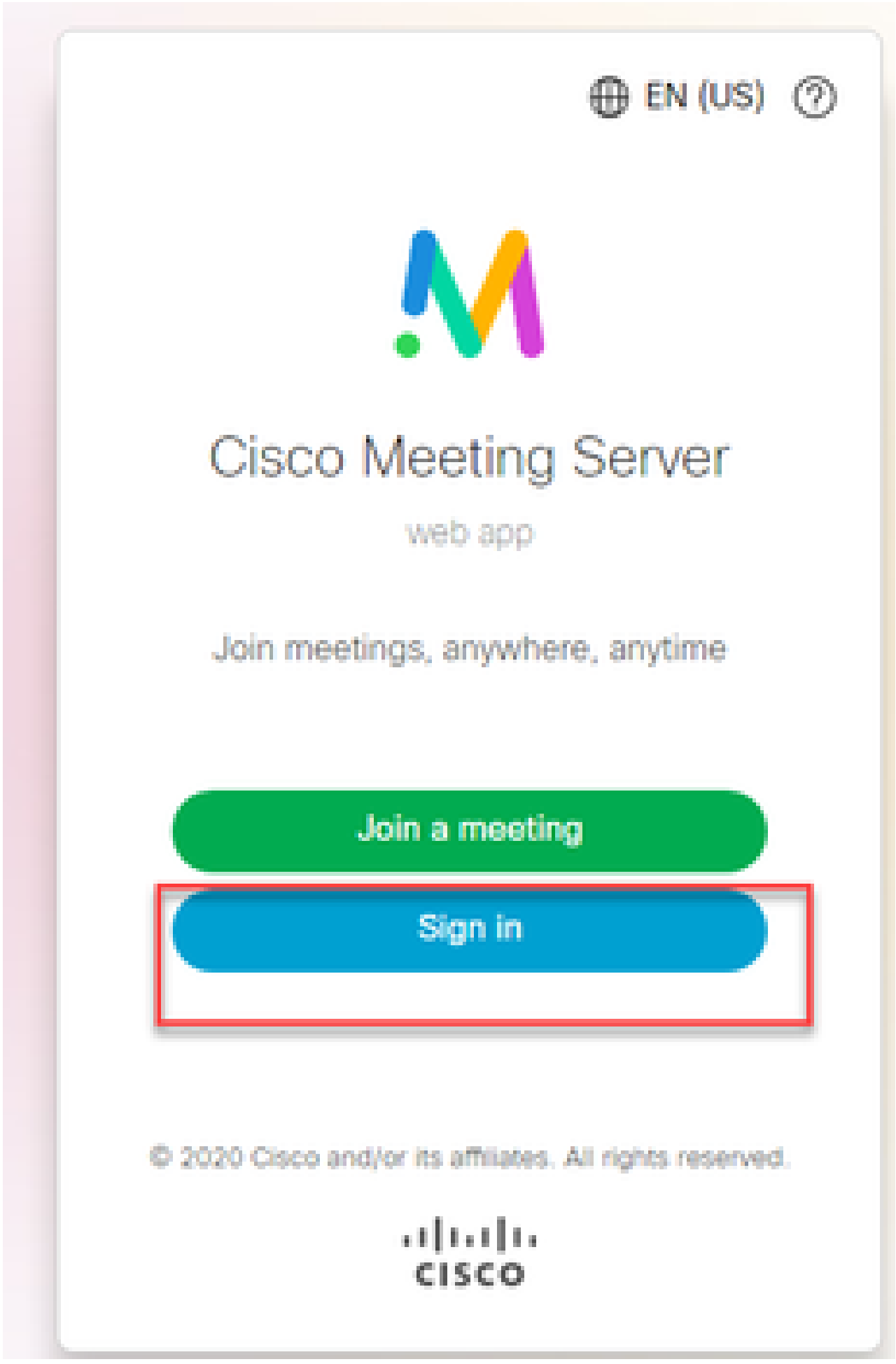
Here, the claim rule shows it is going to send $userPrincipalName$ back to CMS as the UID.



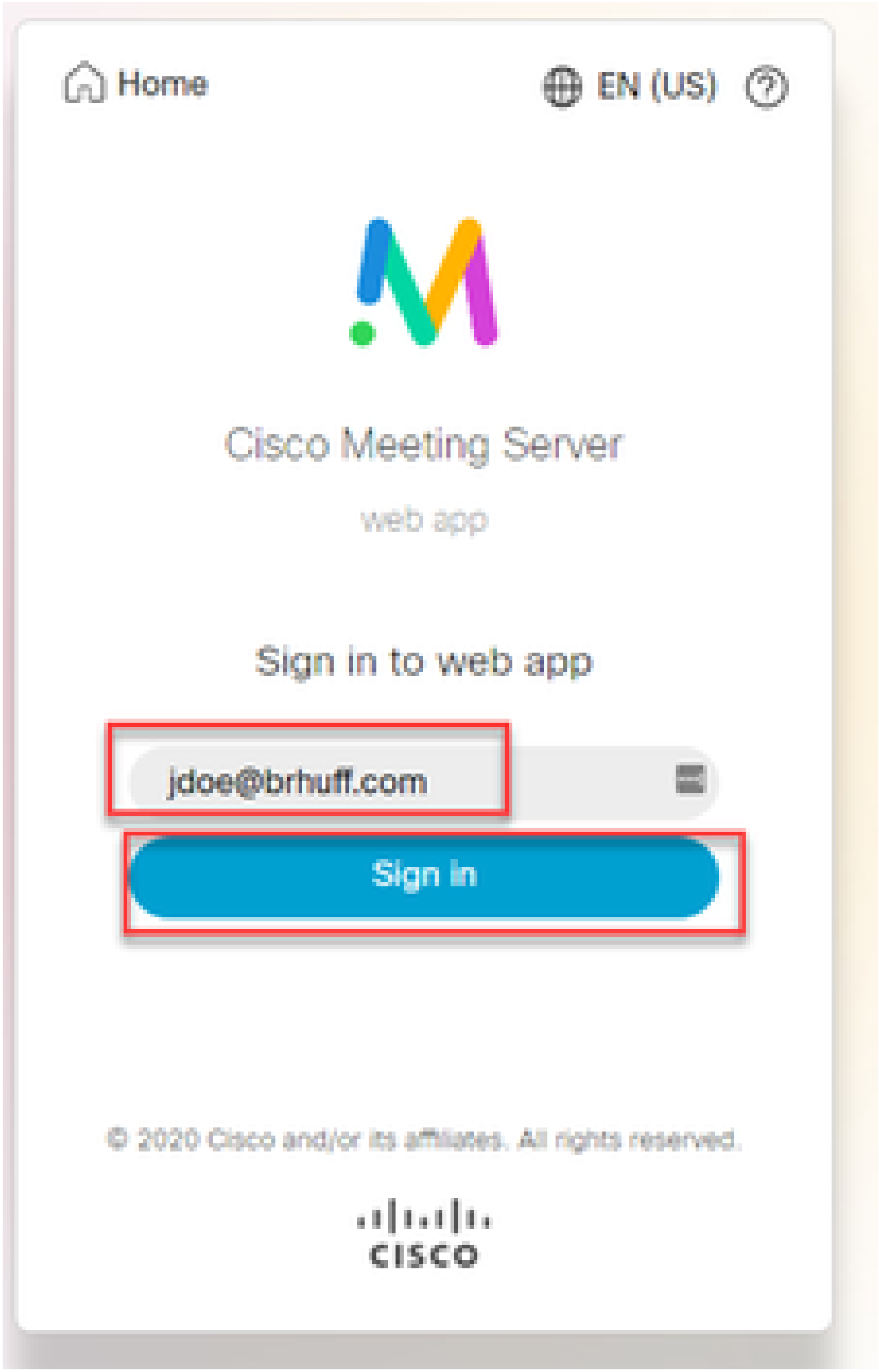
Testing SSO Log in via WebApp

Now that SSO has been configured, you can test the server:

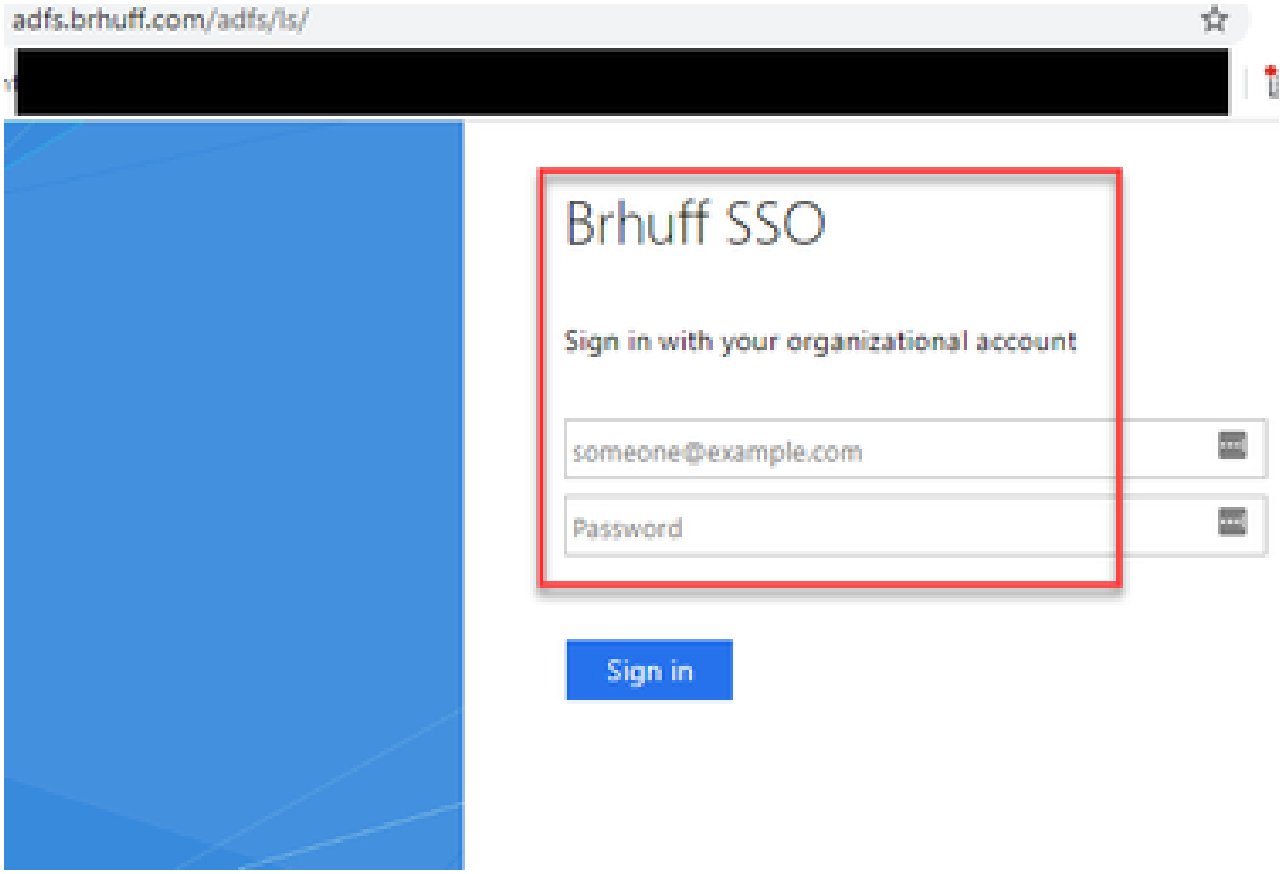
1. Navigate to Webbridge URL for the Web App and select the **Sign in** button.


EN (US)
Cisco Meeting Server
web app
Join meetings, anywhere, anytime
Join a meeting
Sign in
© 2020 Cisco and/or its affiliates. All rights reserved.
cisco

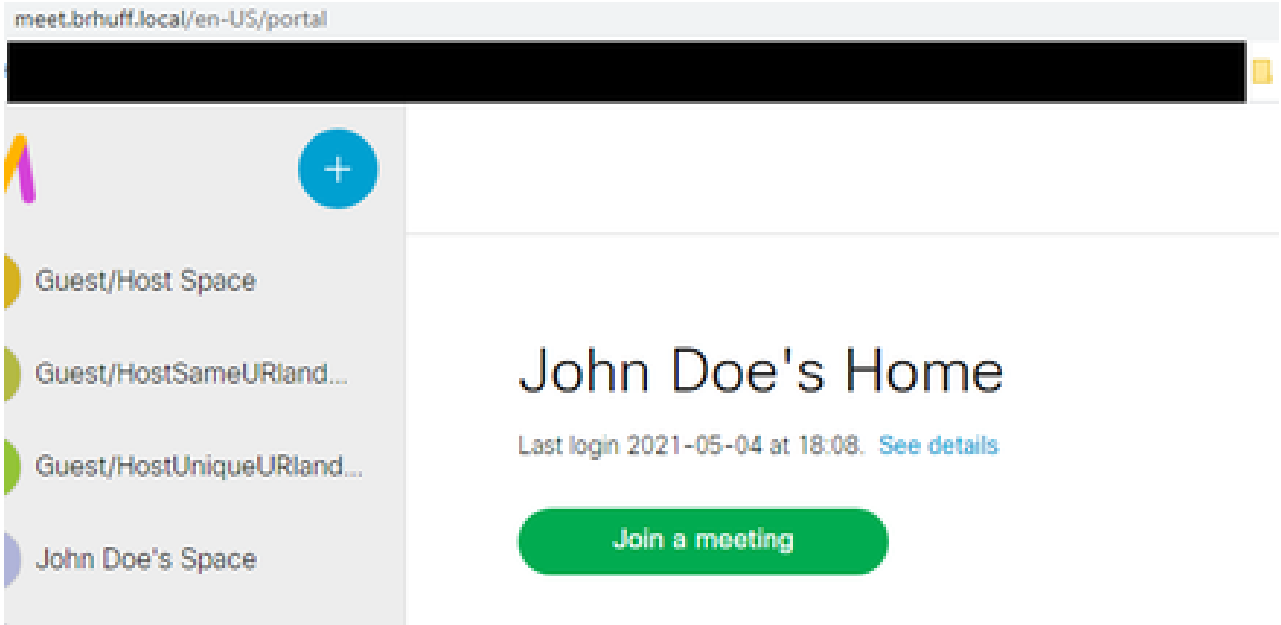
2. The user is presented with the option to input their user name (notice no password option on this page).


Home
EN (US)
Cisco Meeting Server
web app
Sign in to web app
jdoe@brhuff.com
Sign in
© 2020 Cisco and/or its affiliates. All rights reserved.
CISCO

3. The user is then redirected to the ADFS page (after inputting user details) where the user must input their credentials to authenticate to IdP.



4. The user, after inputting and validating credentials with the IdP is redirected with the token to access the Web App home page:



Troubleshooting

Basic Troubleshooting

For basic troubleshooting of any SSO issue:

1. Ensure that the constructed Metadata for the Webbridge3 used to import as a Relying Trust in IdP is configured correctly and the URL configured matches exactly as the ssoServiceProviderAddress in the config.json.
2. Ensure the metadata provided by the IdP and zipped into the Webbridge3 sso configuration file is the latest from the IdP, as if there were any changes to the server host name, certificates, and so on, it needs to be re-exported and zipped into the configuration file.
3. If using signing and encrypting private keys to encrypt data, ensure that the correct matching keys are part of the sso_xxxx.zip file you uploaded to webbridge. If possible, attempt to test without the optional private keys to see if SSO works without this encrypted option.
4. Ensure that the config.json is configured with the correct details for SSO domains, Webbridge3 URL AND expected authenticationmapping to match from the SAMLResponse.

It would also be ideal to attempt the troubleshooting from the log perspective:

1. When navigating to the Webbridge URL, place **?trace=true** at the end of the URL to enable a verbose logging on the CMS syslog. (ex: https://join.example.com/en-US/home?trace=true).
2. Run the **syslog follow** on the Webbridge3 server to capture live during testing or run the test with the trace option appended to the URL and collect the logbundle.tar.gz from the Webbridge3 and CMS Callbridge servers. If webbridge and callbridge are on the same server, this requires only the single logbundle.tar.gz file.

Microsoft ADFS failure codes

Sometimes, there is a failure for the SSO process that can result in a failure for the IdP configuration or its communication with the IdP. If using the ADFS, it would be ideal to review the next link to confirm the failure being seen and take remediation action:

Microsoft Status codes

An example of this is:

> client_backend: ERROR : SamlManager : SAML Authentication request _e135ca12-4b87-4443-abe1-30d396590d58 failed with reason: urn:oasis:names:tc:SAML:2.0:status:Responder

This error indicates that per the previous documentation, the failure occurred due to the IdP or ADFS and thus required to be handled by the Administrator of the ADFS to resolve.

Failed to obtain authenicationID

There can be instances in which during the exchange of SAMLResponse back from the IdP, the Webbridge can display this error message in the logs with a failure in logging in via SSO:

> client_backend: INFO : SamlManager : [57dff9e3-862e-4002-b4fa-683e4aa6922c] Failed obtaining an authenticationId

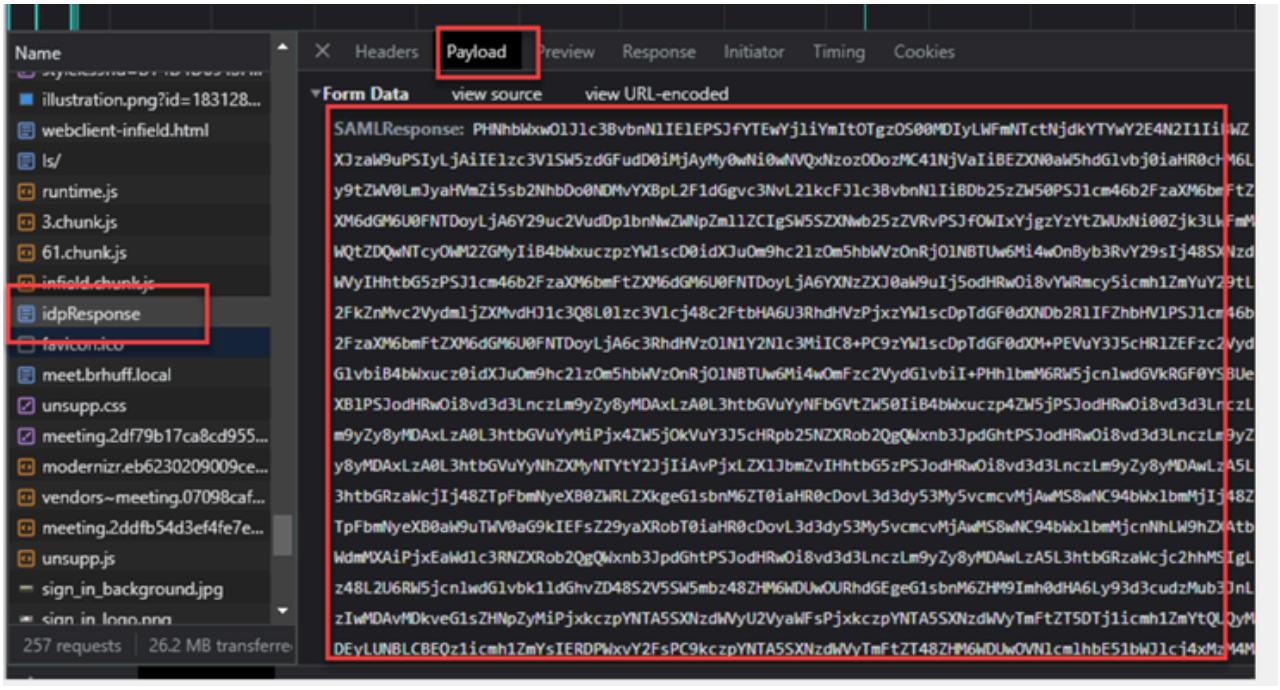
What this indicates is that when reviewing the SAMLResponse data passed back from the IdP during the authentication exchange, the Webbridge3 did not find a valid matching attribute in the response compared to its config.json for the **authenticationId**.

If the communication is not encrypted with the use of the sign and encryption private keys, the SAML Response can be extracted from the Developer Tools Network Logging via a web browser and decoded using base64. If the response is encrypted, you can request the decrypted SAML response from the IdP side.

When recording network logs, ensure the check box for "Preserve Log" is checked, so that the logs are not overwritten when the page changes.



In the developer tools network logging output, also referred to as the HAR data, look for idpResponse under the name column and select **Payload** to see the SAML response. As mentioned previously, this can be decoded using base64 decoder.



When receiving the SAMLResponse data, check the section of **<AttributeStatement>** to locate the attribute names sent back and within this section you can find the claim types configured and sent from the IdP. For example:

> <AttributeStatement>
> <Attribute Name="<URL for commonname">
> <AttributeValue>testuser1</AttributeValue>
> </Attribute>

<Attribute Name="<URL for NameID">
<AttributeValue>testuser1</AttributeValue>
</Attribute>
<Attribute Name="uid">
<AttributeValue>testuser1</AttributeValue>
</Attribute>
</AttributeStatement>

Reviewing the previous names, you can check the **<AttributeName>** under the Attribute Statement section and compare each value to what is set in the **authenticationIdmapping** section of the SSO **config.json**.

In the previous example, you can see that configuration for the **authenticationIdMapping** does NOT match exactly what is passed and thus results in the failure to locate a matching authenticationId:

authenticationIdMapping : http://example.com/claims/NameID

In order to resolve this issue, there are two possible methods to attempt:

1. The IdP Outgoing claim rule can be updated to have a matching claim that matches exactly what is configured in **authenticationIdMapping** of the config.json on the Webbridge3. (Claim rule added on IdP for http://example.com/claims/NameID)
   OR
2. The config.json can be updated on the Webbridge3 to have the 'authenticationIdMapping' matching exactly what is configured as one of the Outgoing claim rules configured on the IdP. (That is '**authenticationIdMapping**' to be updated to match one of the attribute names, which could be "uid", "<URL>/NameID", or "<URL>/CommonName". As long as it matches (exactly) the expected value configured on the Callbridge API when passed)

No assertion passed/matched in validation

Sometimes, during the exchange of the SAMLResponse from the IdP, the Webbridge displays this error indicating there is a failure in matching the assertion and skips any assertions that do not match the server configuration:

client_backend: ERROR : SamlManager : No assertions passed the validation
client_backend: INFO : SamlManager : Skipping assertion without us in the allowed audience

What this error indicates is that when reviewing the SAMLResponse from the IdP, the Webbridge failed to locate any matching assertions and thus skipped non-matching failures and ultimately resulted in a failured SSO log in.

In order to locate this issue, it is ideal to review the SAMLResponse from the IdP. If the communication is not encrypted with the use of the sign and encryption private keys, the SAML Response can be extracted from the **Developer Tools Network Logging** via a web browser and decoded using base64. If the response is encrypted, you can request the decrypted SAML response from the IdP side.

When reviewing the SAMLResponse data, looking at the **<AudienceRestriction>** section of the response, you can find all audiences that this response is restricted for:

<Conditions NotBefore=2021-03-30T19:35:37.071Z NotOnOrAfter=2021-03-30T19:36:37.071Z>
<AudienceRestriction>

<Audience>https://cisco.example.com</Audience>
</AudienceRestriction>
</Conditions>

Using the value in the **<Audience>** section (https://cisco.example.com) you can compare it to the **ssoServiceProviderAddress** in the config.json of the Webbridge configuration and validate if it is an exact match. For this example, you can see that reason for the failure is the Audience does NOT match the Service provider address in the configuration, because it has the appended **:443**:

ssoServiceProviderAddress : https://cisco.example.com:443

This requires an exact match between these to not result in a failure such as this. For this example. the fix would be to either of these two methods:

1. The **:443** could be removed from the address in the ssoServiceProviderAddress section of the config.json, so that it matches the **Audience** field provided in the SAMLResponse from the IdP.
OR

2. The metadata OR relying trust party for Webbridge3 in the IdP can be updated to have the **:443** appended to the URL. (If the metadata is updated, it must be imported again as a **Relying Trust Party** on the ADFS. However, if you modify the Relying Trust Party from the IdP wizard directly, it does not need to be imported again.)

Also, pay attention to the condition NotBefore and NotONOrAfter: <Conditions **NotBefore**=2021-03-30T19:35:37.071Z **NotOnOrAfter**=2021-03-30T19:36:37.071Z>

If the server time is incorrect, it can cause the time to fall out of the validity period defined in the conditions. You will want to check the NTP servers via the CLI using commands **ntp server list** to review the configured NTP servers and **ntp status** to check the status of the configured NTP servers. Use the command **date** to verify the servers time.

Tip: If using a local/internal NTP server, try configuring a public NTP server such as time.google.com (make sure to configure a public DNS server before).

Sign in Failed on Web App:

Home

EN (US)



Blahman Industries

Blahman WebApp

Sign in to web app

darmckin@brhuff.com

Sign In

Sign in failed

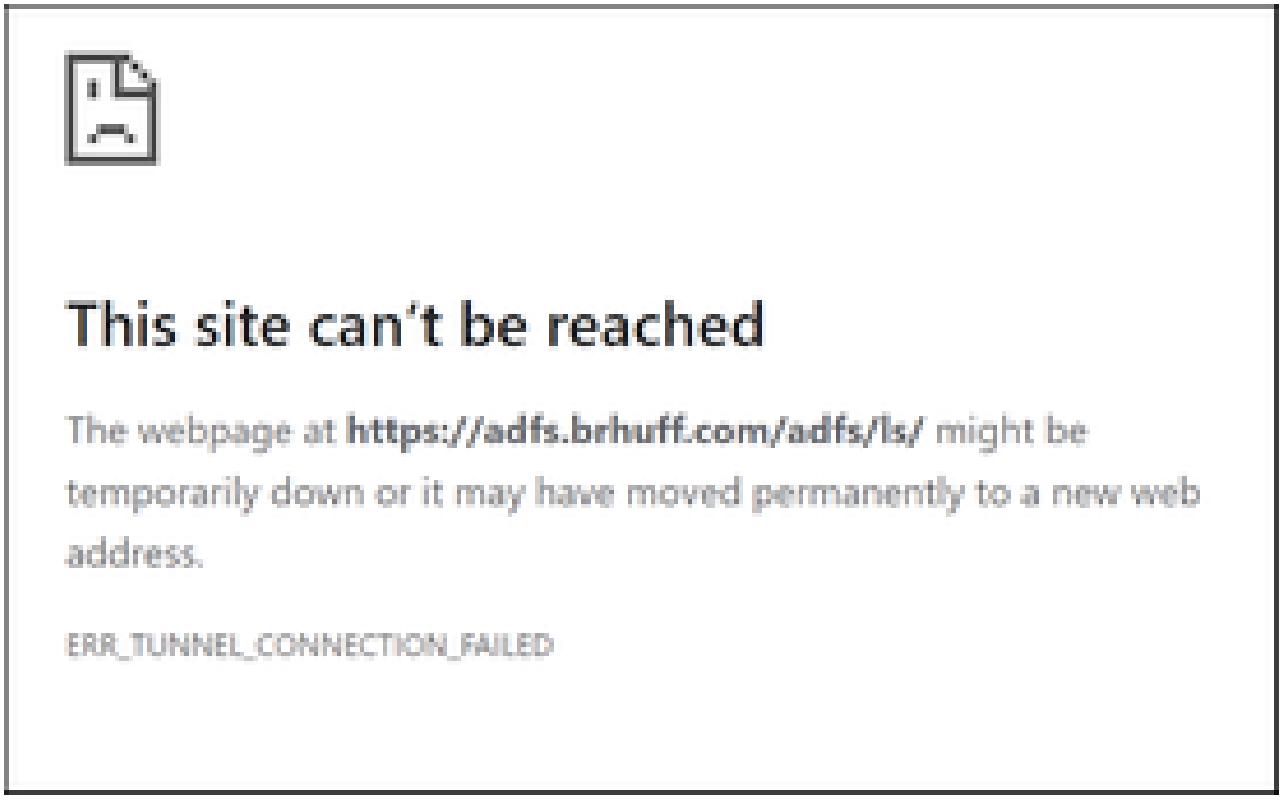
© 2019-2023 Cisco and/or its affiliates. All rights reserved.

CISCO

CMS FAQ

Cisco Website

), webbridge checks that the domain used matches one in the config.json file, then sends the SAML information to the client, telling the client where to connect to for authentication. The client attempts to connect to the IdP that is in the SAML token. In the example, the browser shows this page because it cannot reach the ADFS server.



Error on Client browser

CMS Webbridge traces (while ?trace=true is used)

Mar 19 10:47:07.927 user.info cmscb3-1 client_backend: INFO : SamlManager : [63cdc9ed-ab52-455c-8bb2-9e925cb9e16b] Matched SSO sso_2024.zip in SAML Token Request

Mar 19 10:47:07.927 user.info cmscb3-1 client_backend: INFO : SamlManager : [63cdc9ed-ab52-455c-8bb2-9e925cb9e16b] Attempting to find SSO in SAML Token Request

Mar 19 10:47:07.930 user.info cmscb3-1 client_backend: INFO : SamlManager : [63cdc9ed-ab52-455c-8bb2-9e925cb9e16b] Successfully generated SAML Token

Scenario 2:

User attempted to sign in using domain that is not in the SSO zip file on webbridge signin page. Client sends in a tokenRequest with a payload of the username the user entered. Webbridge stops the login attempt immediately.

CMS Webbridge traces (while ?trace=true is used)

Mar 18 14:54:52.698 user.err cmscb3-1 client_backend: ERROR : SamlManager : Invalid SSO login attempt

Mar 18 14:54:52.698 user.info cmscb3-1 client_backend: INFO : SamlManager : [3f93fd14-f4c9-4e5e-94d5-49bf6433319e] Failed finding an SSO in SAML Token Request

Mar 18 14:54:52.698 user.info cmscb3-1 client_backend: INFO : SamlManager : [3f93fd14-f4c9-4e5e-94d5-49bf6433319e] Attempting to find SSO in SAML Token Request

Scenario 3:

User has entered the correct username and is presented the SSO sign in page. The user enters the correct username and password here too, but still gets Sign in Failed

CMS Webbridge traces (while ?trace=true is used)

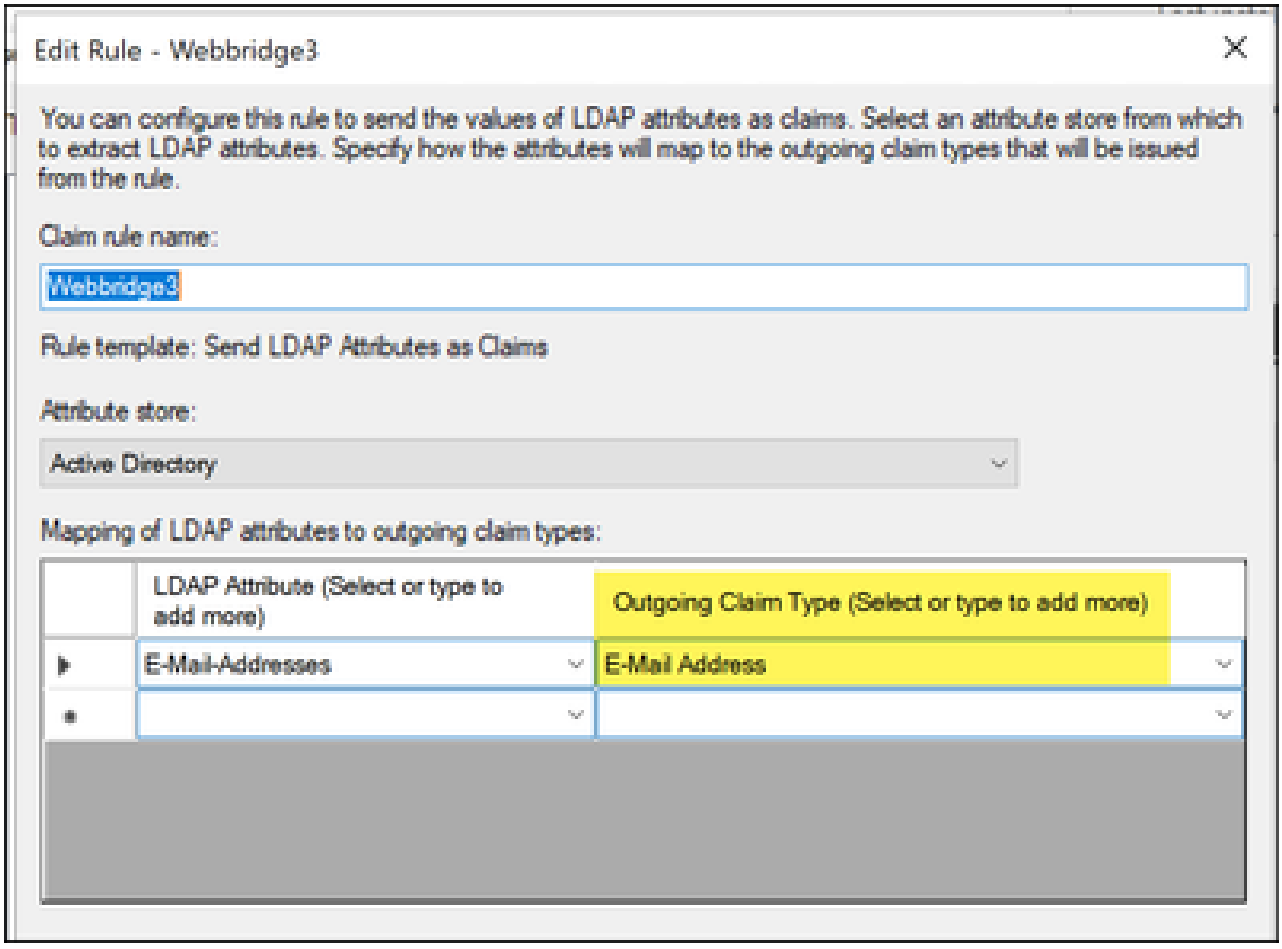
Mar 19 16:39:17.714 user.info cmscb3-1 client_backend: INFO : SamlManager : [ef8fe67f-685c-4a81-9240-f76239379806] Matched SSO sso_2024.zip in SAML Token Request

Mar 19 16:39:17.714 user.info cmscb3-1 client_backend: INFO : SamlManager : [ef8fe67f-685c-4a81-9240-f76239379806] Attempting to find SSO in SAML IDP Response

Mar 19 16:39:17.720 user.err cmscb3-1 client_backend: ERROR : SamlManager : No authenticationId mapped element found in signed SAML Assertions

Mar 19 16:39:17.720 user.info cmscb3-1 client_backend: INFO : SamlManager : [ef8fe67f-685c-4a81-9240-f76239379806] Failed obtaining an authenticationID

The cause for scenario 3 was the claim rule in the IdP was using a claim type that did not match the authenticationIdMapping in the config.json file used in the SSO zip file that was uploaded to webbridge. Webbridge is looking at the SAML response and expects the attribute name to match what is configured in the config.json.



Claim Rule in ADFS

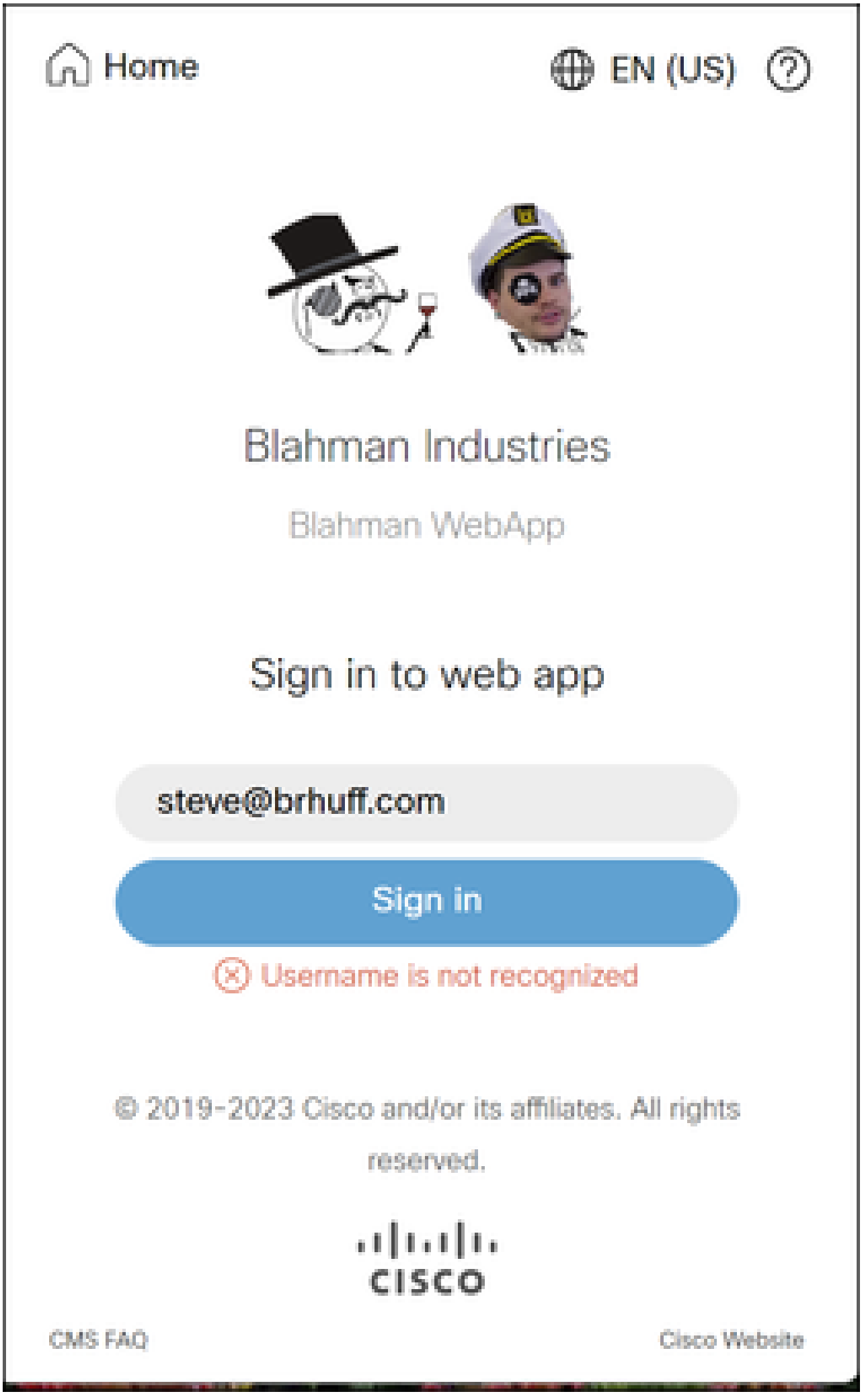


config.json example

Username is not Recognized

Scenario 1:

User signed in with wrong username (Domain matches what is in the SSO zip file that was uploaded to webbridge3, but user does not exist)



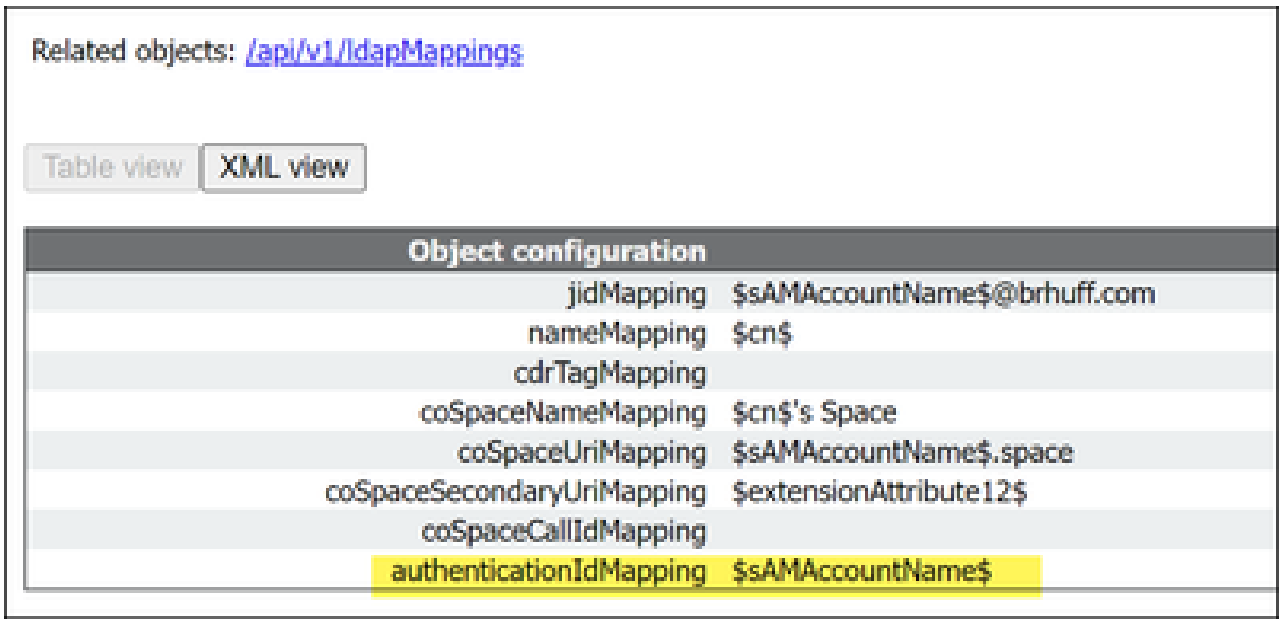
Username is not recognized

in CMS ldapmapping does not match the configured LDAP attribute used for claim rule in ADFS. The line saying "**Successfully obtained authenticationID:darmckin@brhuff.com**" is saying ADFS has claim rule configured with attribute that gets darmckin@brhuff.com from active directory, but the AuthenticationID in CMS **API** > **Users** shows it is expecting darmckin. In the CMS ldapMappings, the **AuthenticationID** is configured as **$sAMAccountName$**, but the claim rule in ADFS is configured to send the **E-Mail-Addresses**, so this does not match.

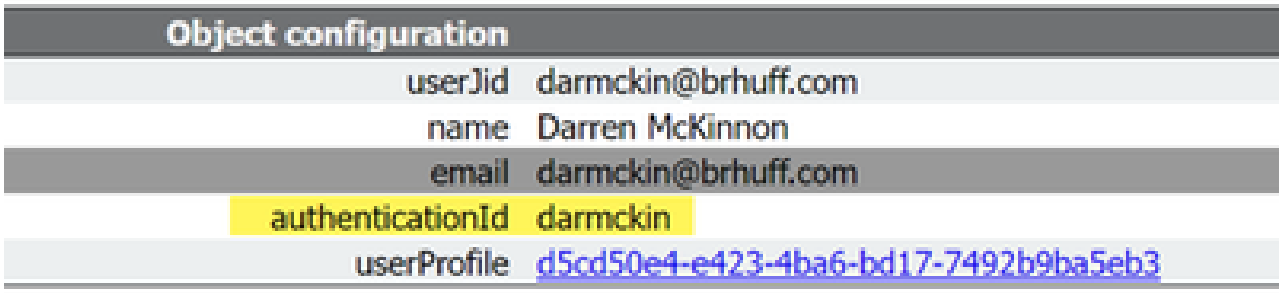
How to fix this:

Do either option to mitigate:

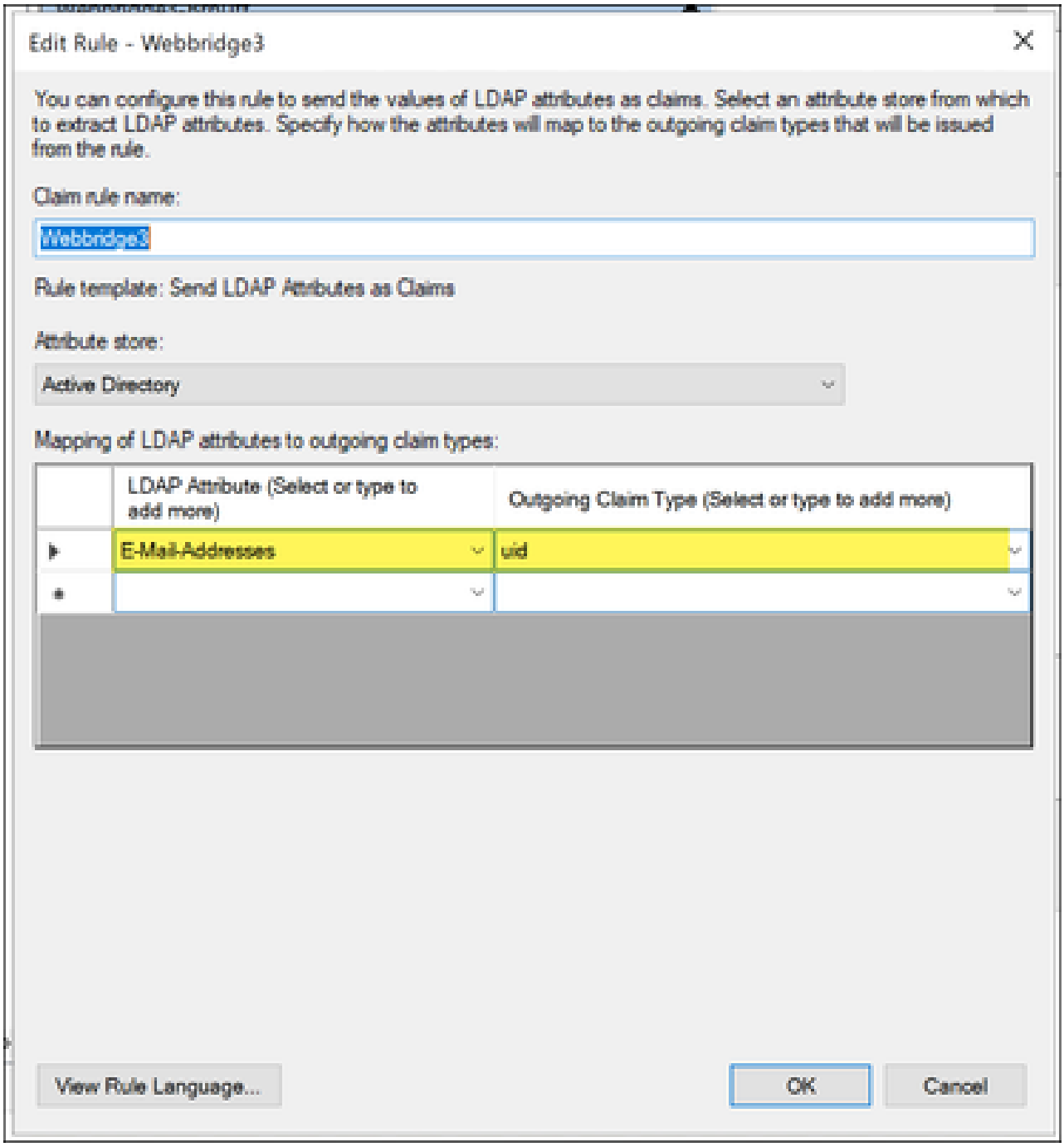
1. Change the AuthenticationID in the CMS ldapmapping to match what is used in the Claim rule on ADFS and perform a new sync
2. Change the LDAP Attribute used in ADFS Claim rule to match what is configured in CMS ldapmapping



API LDAPMapping



API User example



Claim Rule from ADFS

Webbridge log showing working log in example. Example generated using ?trace=true in the join URL:

Mar 18 14:24:01.096 user.info cmscb3-1 client_backend: INFO : SamlManager : [7979f13c-d490-4f8b-899c-0c82853369ba] Matched SSO sso_2024.zip in SAML Token Request

Mar 18 14:24:01.096 user.info cmscb3-1 client_backend: INFO : SamlManager : [7979f13c-d490-4f8b-899c-0c82853369ba] Attempting to find SSO in SAML IDP Response

Mar 18 14:24:01.101 user.info cmscb3-1 client_backend: INFO : SamlManager : [7979f13c-d490-4f8b-899c-0c82853369ba] Successfully obtained authenticationID:darmckin@brhuff.com

Mar 18 14:24:01.102 user.info cmscb3-1 host:server: INFO : WB3Cmgr: [7979f13c-d490-4f8b-899c-0c82853369ba] AuthRequestReceived for connection id=64004556-faea-479f-aabe-

691e17783aa5 registration=40a4026c-0272-42a1-b125-136fdf5612a5 (user=darmckin@brhuff.com)

Mar 18 14:24:01.130 user.info cmscb3-1 host:server: INFO : successful login request from darmckin@brhuff.com

Mar 18 14:24:01.130 user.info cmscb3-1 host:server: INFO : WB3Cmgr: [7979f13c-d490-4f8b-899c-0c82853369ba] issuing JWT ID e2a860ef-f4ef-4391-b5d5-9abdfa89ba0f

Mar 18 14:24:01.132 user.info cmscb3-1 host:server: INFO : WB3Cmgr: [7979f13c-d490-4f8b-899c-0c82853369ba] sending auth response (jwt length=1064, connection=64004556-faea-479f-aabe-691e17783aa5)

Mar 18 14:24:01.133 local7.info cmscb3-1 56496041063b wb3_frontend: [Auth:darmckin@brhuff.com, Tracing:7979f13c-d490-4f8b-899c-0c82853369ba] 10.10.10.8 - - [18/Mar/2024:18:24:01 +0000] status 200 "POST /api/auth/sso/idpResponse HTTP/1.1" bytes_sent 0 http_referer "https://adfs.brhuff.com/" http_user_agent "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0. Safari/537.36" to upstream 192.0.2.2:9000: upstream_response_time 0.038 request_time 0.039 msec 1710786241.133 upstream_response_length 24 200

Frequently Asked Questions (FAQ)

This section highlights frequently asked quesitons or topics pertaining to WebApp SSO on the Cisco Meeting Server.

Can the Webapp SSO JWT be extended?

The JSON Web Token (JWT) is the token that is provided by the Callbridge to a successfully authenticated Webapp client (successfully authenticated to the IdP), granting them access to the WebApp services. Within the JWT is an expiry timer value that indicates how long the JWT is valid, which once the JWT expiry time is reached - the WebApp user is redirected back to the Webbridge login in page, requiring to reauthenticate in order to get access back.

The JWT expiry is configurable utilizing the **'callbridge wc3jwt expiry <1-24>'** (Added in 3.8 and later - more details can be found in the CMS 3.8 Release notes or CMS MMP Guide) in which the numeric value is for the number of hours you want to set the expiry time for the JWT provided to WebApp clients. However, as seen in the command, the max value can be set is **24 hours**, which means that the longest time the JWT can remain valid and WebApp user can logged in is 24 hours. When the JWT expires time is met - the browser dumps the expired token and the WebApp user is forced back to the WebApp login page.

In some environments, depending on the IdP and environment setup, a Persistent SSO/Keep me signed in features can be implemented on the IdP - which would provide the browser with a persistent cooked from the IdP, where even closing the browser, the cookie would be retained (unless cleared by other means). **Regardless of the SSO/IdP configuration - when the JWT expires (max 24 hours), the WebApp user is forced back to the WebApp login page - however, in this scenario where Persistent SSO is enabled on the IdP - the user would just need to input their <user@domain> on the WebApp login page and not need to re-authenticate to their IdP.**

An enhancement is open for implementing a Refresh token mechanism to allow for extended authorization to WebApp - Cisco bug ID CSCwm28758.

Do I need to re-authenticate to WebApp if I close my browser?

The flow for this scenario would be:

1. A user logs into the WebApp (using SSO authentication method).
2. The user closes the browser at some point.
3. The user opens the browser again and navigates to WebApp site. (either immediately or at later time).

What would happen in this scenario?

For this answer it depends! It entirely depends on whether the JWT provided by Callbridge is expired at the time of access the WebApp page. As long as the JWT is still valid and present in storage, you can navigate to the WebApp page (even if you closed the browser). Keep in mind the max amount of time the JWT can be valid is **24 hours.**

- If the JWT is still **VALID (NOT EXPIRED) and Present**, the user can navigate to the WebApp page with their spaces and so on **without needing to re-authenticate**.
- If the JWT is **INVALID (EXPIRED)**, the user is redirected to the WebApp login page, needing to relogin to the WebApp. The user could also have to re-authenticate to their IdP depending on their IdP authentication method (for example if the IdP is using Session SSO or Persistent SSO).

How are multiple domains supported in WebApp SSO?

WebApp SSO is capable of supporting environments that have multiple domains and even environments where those different domains point to different Identity Providers (IdPs). Please review the Cisco Meeting Server deployment guides or contact Cisco TAC for support on using multiple domains.

Related Information

- Cisco Technical Support & Downloads