Deploy Content Services GraphQL into a traditional IBM WebSphere Application Server environment

The IBM Content Services GraphQL (CSGQL) API is a part of the IBM® FileNet® P8 suite of products which contain a set of robust APIs that range from core platform APIs to supporting application APIs.

The CSGQL API provides a schema and an easy-to-understand query language system that simplifies application development for your Content Platform Engine (CPE). The API schema definition of types and fields matches Content Engine Java API object model closely, with necessary and desirable extensions for natural GraphQL developer consumption.

This document describes the deployment and configuration of IBM Content Services GraphQL API into a traditional IBM WebSphere Application Server (tWAS) 9.0 environment. With this configuration, CSGQL can communicate with the FileNet Content Manager hosted in tWAS environments. Mixing of either service deployed in a containerized environment is not supported.

V5.5.7a

Roadmap

Prerequisites

Manual versus automation helper scripts for configuration and deployment

Authentication choices

Overview of options for Federated repositories and LDAP servers configuration

CPE information
- CPE LTPA keys export
- CPE certificates export
- CPE connection information

Prepare

Install CSGQL war file and Content Engine JAVA API client files

(Optional) Clone deployment helper scripts to a local repository

Gather federated repositories information
- Option 1: Manually configure CPE and CSQL matching federated repositories
- Option 2: Create CSGQL Federated repositories configuration using CPE CMUI profile
- Option 3: Use deployment helper scripts to create CSGQL federated repositories

Deploy Content Services GraphQL

Deploy war file

Add and configure required shared libraries
- Create shared Libraries
- Associating the shared library with the GraphQL application
- Set the parent last class loader policy

Configure the CS-GraphQL application

Map Application Security to all authenticated

Set JVM arguments

Enable Single Sign On
- LTPA keys import
- Import LTPA keys
- Configure inbound trusted realms

Configure Secure (SSL) communication between CPE and GraphQL

Validate the Configuration

Debug

Trace flags

Success

(Optional) Configure OAuth/OIDC between CS-GraphQL and CPE

Register GraphQL with Identity Provider

XSRF(CSRF)/CORS headers

Prerequisites

These instructions assume other services required in the system are already deployed and configured. These other services include:

- WebSphere traditional application server or cluster environments to host the CPE and CSGQL deployments.
- FileNet Content Manager Content Platform Engine pre-exists at the same version as CSGQL to be deployed.
- Directory Services (LDAP) or Identity Provider (IdP) to manage user authentication.

As part of the preparation, necessary information about these other services must be gathered. The reader of this document is assumed to be knowledgeable about all these services and have the privileges needed to interact with them as needed.

Manual versus automation helper scripts for configuration and deployment

Limitations where manual process should be followed.

- Scripts are written for the Linux/Unix platform only. If the traditional WebSphere Application Server (tWAS) for hosting CSGQL is installed on Windows, the automated bash scripts cannot be used.
- Scripts target deployment under a tWAS single application server environment. Deployment to a tWAS cluster should follow the manual process.
- Configuration of OAUTH is outside of the scope of the scripts. The scripts will configure GraphQL to use BASIC authentication with the Content Platform Engine (CPE) server. Information about OAUTH/OIDC configuration between CSGQL and CPE provided in the below section (Optional) Configure OAuth/OIDC between CS-GraphQL and CPE .

The main automation script will read key-value pairs from an input file and uses the values to invoke subscripts. Each subscript performs one specific task. You can run a subscript directly by passing in the same input file. The subscript will access the key-value pairs required to complete its particular task.

After reviewing these manual instructions and the scripts, you may decide a combination of the two better fits your situation.

Authentication choices

The documented steps follow the BASIC authentication configuration first to establish a base working environment. Those steps are followed with the additional configuration needed to support single sign on (SSO) integrated with an IdP. In the production environment, it is recommended to utilize CSGQL with SSO only. BASIC authentication is not recommended for production use.

Overview of options for Federated repositories and LDAP servers configuration

To support CSGQL deployment in tWAS, the CPE and CSGQL applications must be deployed into WebSphere instances configured to use federated LDAP repositories. The applications can be deployed within the same tWAS cell or on the same tWAS node. But they must not share a tWAS application server instance since each needs to run in a distinct JVM.

If the applications will be deployed into different tWAS cells, to avoid issues with authentication and communication, the LDAP configuration created in the tWAS instance hosting CSGQL must be identical to the ldap configuration for the tWAS instance hosting the CPE. Three options exist to avoid trouble:

Option 1: Manually configure CPE and GSQGL matching federated repositories
Option 2: Create CSGQL Federated repositories using CPE CMUI profile
Option 3: Use deployment helper scripts to create CSGQL federated repositories

You will need to gather LDAP servers and configuration information needed from CPE tWAS to populate or verify matching security on GraphQL tWAS server. Minimally this information is needed for each LDAP server included as a federated LDAP repository:

Directory type
Primary host name and port
Bind distinguished name and Bind password
User, group, and group membership attributes
Login property

For details of where the required information will be used, see section Add each LDAP server configuration as a LDAP repository below in the Procedures.

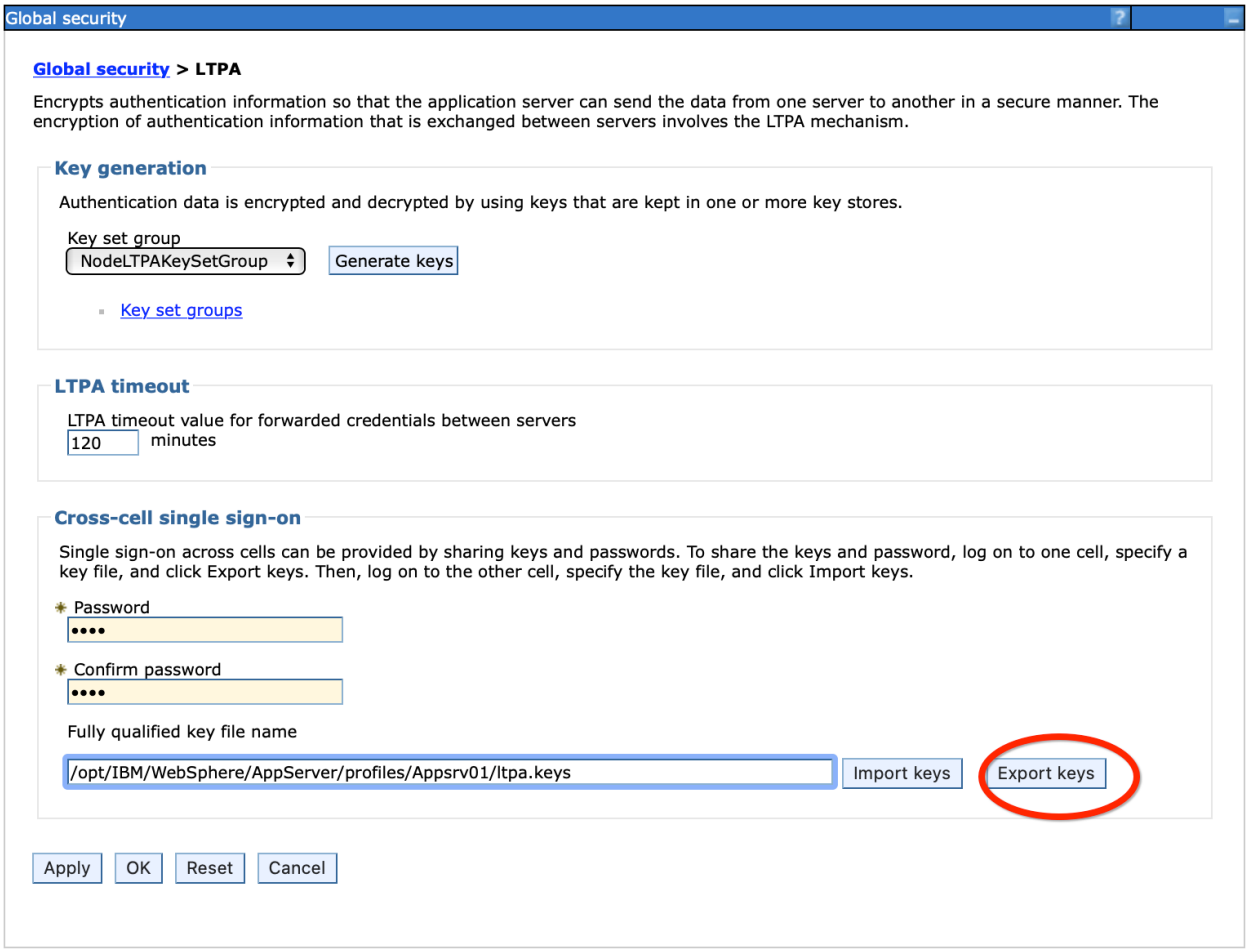
CPE information

Complete the following tasks to export or gather information about the configuration of the tWAS environment hosting the CPE. The information is used either to complete the configuration tasks manually or as inputs to the configureGQL scripts.

CPE LTPA keys export

Export CPE LTPA key from tWAS and download to a system with access to the tWAS instance targeted for the CSGQL deployment.

1. In the CPE tWAS administration console, navigate to **Security > Global security > LTPA**.
2. Provide a password for the ltpa.keys file. Remember this password
3. Provide a path for the LTPA file, for example, /opt/IBM/WebSphere/AppServer/profiles/Appsrv01/ltpa.keys
4. Click **Export keys**



CPE certificates export

(Optional) If communications between the CPE and CSGQL will not utilize SSL, this step is not required. If SSL will be used and the CPE certificate is signed by a trusted root authority, this step is not required if the tWAS environment for CSGQL includes the trusted root authority trust certificate.

If SSL communications will use a self-signed certificate to secure communications between the CSGQL server and the CPE server, the SSL certificate for the CPE must be exported from the CPE application server. Export certificates using to .pem format file.

To export using command-line tools:

Example commands to export SSL certificate from **CPE** tWAS server keystore

```
cd <APP_SERVER_ROOT>/profiles/AppSrv01/config/cells/<CELL_NAME>/nodes/<NODE_NAME>

<APP_SERVER_ROOT>/java/8.0/bin/keytool -export -alias default -keystore
<KEYSTORE_PATH> -storetype pkcs12 -storepass WebAS -rfc -file cpe.pem
```

To directly retrieve a signer certificate from the tWAS environment for the CPE into the tWAS environment for CSGQL, follow this instructions in this WebSphere Application Server documentation topic:

> Adding the signer certificate from the secondary deployment manager to the local trust store

CPE connection information

You will need the following information to construct the URL for the CPE.

Server	Hostname on which Content Platform Engine is accessed or route to a load-balancer for the CPE
Port	Communication port number for the application server. The port value is optional if the connection does not require a port to specified in the URL
Path	Relative path to the web service interface endpoint for Content Platform Engine. By default, the path is set to `wsi/FNCEWS40MTOM/`

Example -Decm.content.remote.cpeuri=http://xyz.example.com:9080/wsi/FNCEWS40MTOM/

Prepare

Several tasks must be completed before the deployment of the CSGQL API application. These are either performed using the WASt administrative interface or on the machine that has access to the WASt administrative interface.

Install CSGQL war file and Content Engine JAVA API client files

On the machine where CSGQL application will be manually deployed into WASt or the helper scripts deploy the CSGQL application will be run, complete the following actions in the installation wizard.

1. Run the Content Engine Client installer that is from the same release version as the targeted CPE.
2. Read and accept the licenses presented
3. Specify the folder path where the installer will place the files then select Next
4. Provide the CPE connection information from which you intend to download the files
5. Choose as the FileNet P8 applications to install then select Next
    a. "Java Client Application"
    b. "IBM Content Services GraphQL API servlet"
6. As the Application Server Type, select "WebSphere Application Server" from the dropdown then select Next
7. Review the CPE URLs are correct, then select Next
8. Review the CPE URL for the Web Services Transport is correct, then select Next
9. Select Install to initiate the download of the files.
10. Confirm the download was successful be reviewing the files present in the `<installation path>/lib` directory contain the CPE java client APIs files and the `content-graphql-api.war` file.

(Optional) Clone deployment helper scripts to a local repository

Create a local copy of the Git Hub repository that contains the CSGQL deployment automation helper scripts.

1. Download or clone the repository on your local machine:

```
git clone https://github.com/ibm-ecm/ibm-content-platform-engine-samples
```

2. Change to the CSGraphQLAPIDeployScripts folder that supports tWAS in your local repository:

```
cd CSGraphQLAPIDeployScripts/websphere
```

If you plan to utilize the deployment helper scripts, the remainder of this preparation section as well as the section Deploy Content Services GraphQL that follows will be skipped. However the information in this document can be used to understand the information required by the helper scripts to automate these same tasks.

Continue with the instructions in the readme.md file located in the `CSGraphQLAPIDeployScripts/websphere` folder or by accessing the copy in GitHub here. After completing the process described in the readme.md for the deployment helper scripts, return to section in this document Validate the Configuration.

Gather federated repositories information

Before deploying the CSGQL application onto tWAS, verify the LDAP server configuration information gathered during the preparation phase is correct and current. To avoid issues, the LDAP configuration created in the tWAS instance hosting CSGQL must be identical to the ldap configuration for the tWAS instance hosting the CPE. Follow the instructions below to create configurations in the tWAS instance hosting the CSGQL deployment.

For additional information about the CPE and configuration of the directory services in WASt, see Configuring Content Platform Engine application server authentication (LDAP) settings

> Note: If you choose to use the deployment helper scripts, this entire topic can be skipped. However the information here can be used to verify the configuration produced by the helper scripts is correct or to troubleshoot issues.

Option 1: Manually configure CPE and CSQL matching federated repositories

In this step we go through the step-by-step instruction on configuring the ldap as a federated repository in tWAS. If your CPE ldap configuration is different or contains any additional changes, please apply the same to the CSGQL ldap configuration to ensure they are in-sync. The resulting federated repositories configuration in the tWAS for CSGQL contains the same LDAP definitions as a present in the tWAS hosting the CPE/

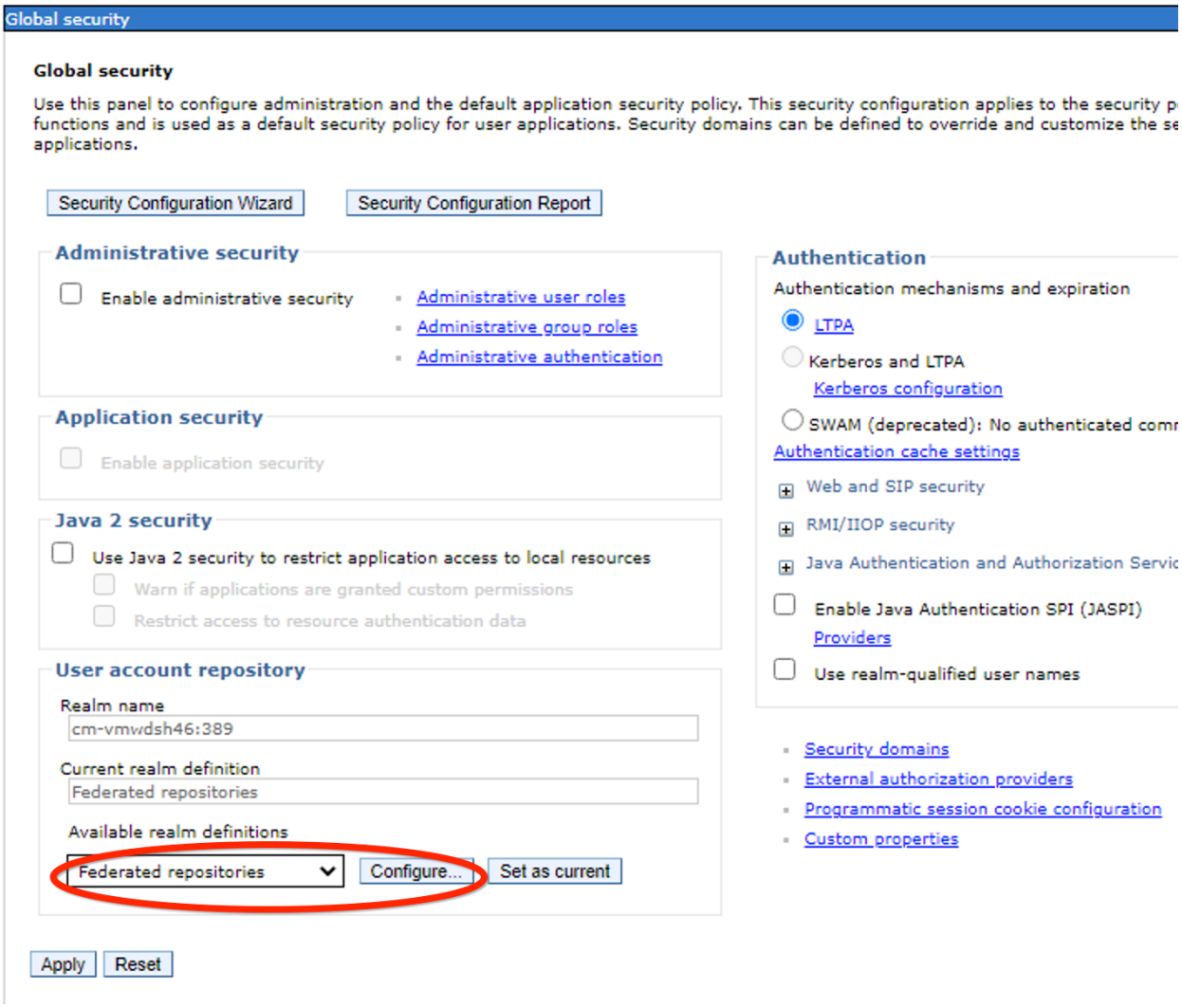
1. Start WebSphere Application Server and log in to the Integrated Solutions Console on the Deployment Manager by going to the following web address: http://websphere_Application_Server_host_name:port/ibm/console

2. Click **Log in** and enter the credentials of the administrative user ID that you specified during the installation of WebSphere Application Server.

3. Click **Security** > **Global Security.**

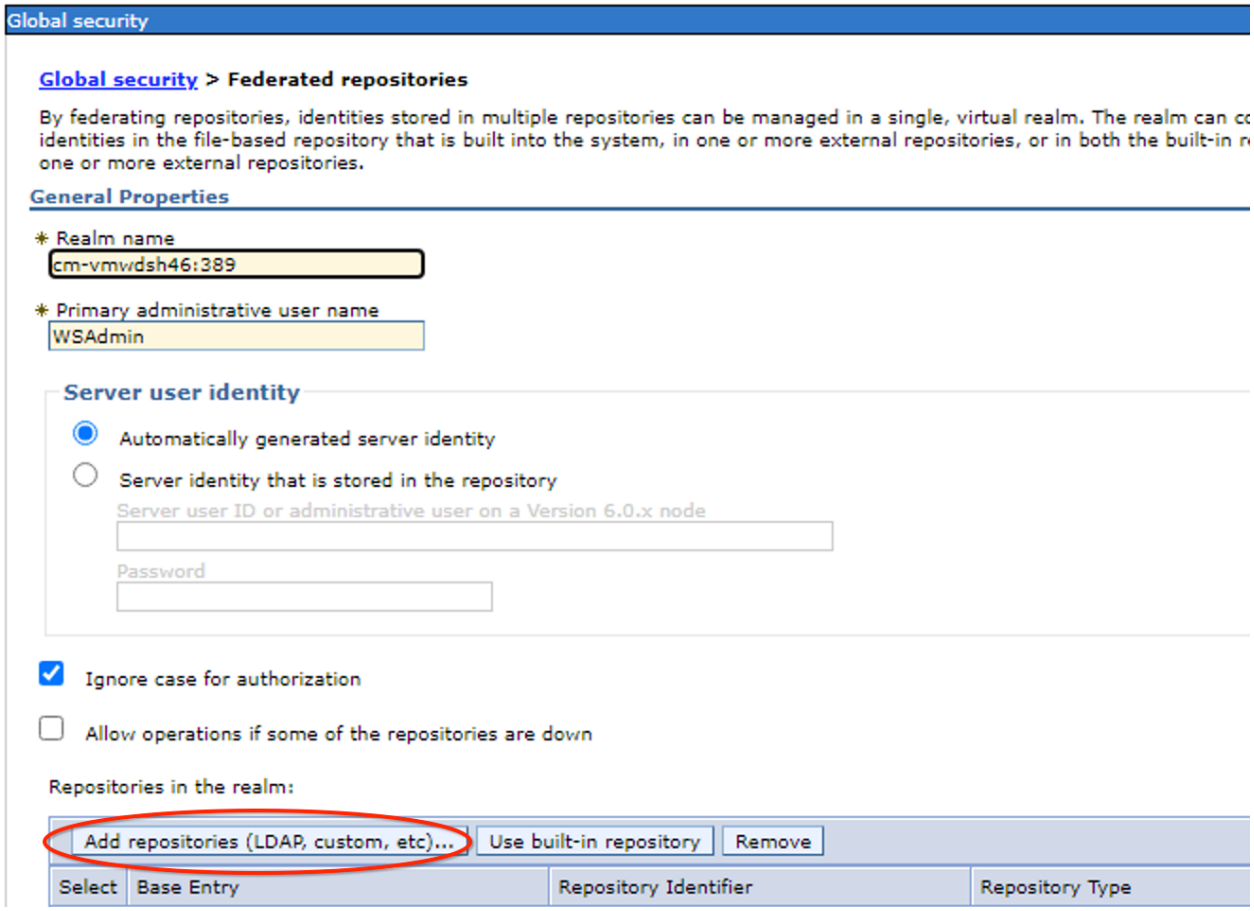
Create Federated Repository entry

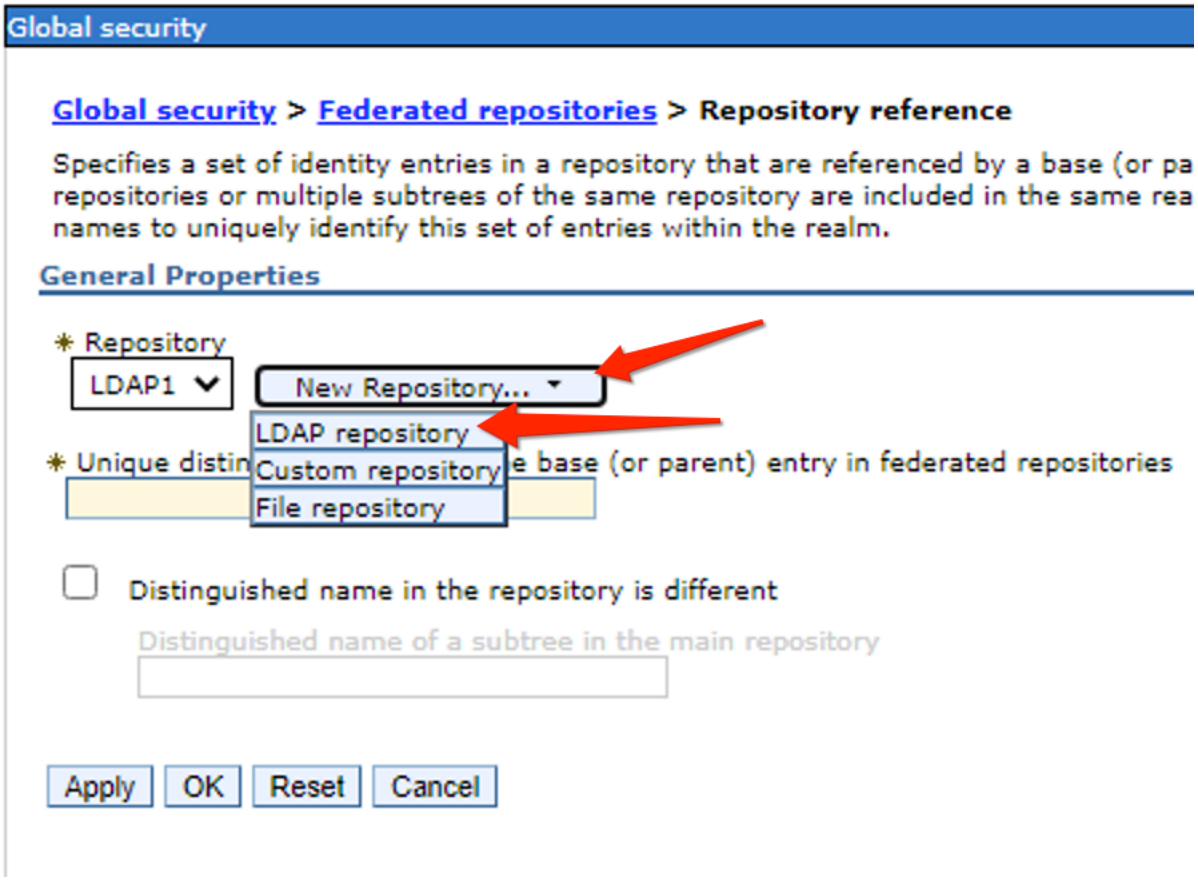
4. Select **Federated Repositories** from the **Available realm definitions** field, and then click **Configure**.



Add each LDAP server configuration as a LDAP repository

5. Click **Add repositories** and then, on the Repository reference page, click **New Repository** > **LDAP repository**.



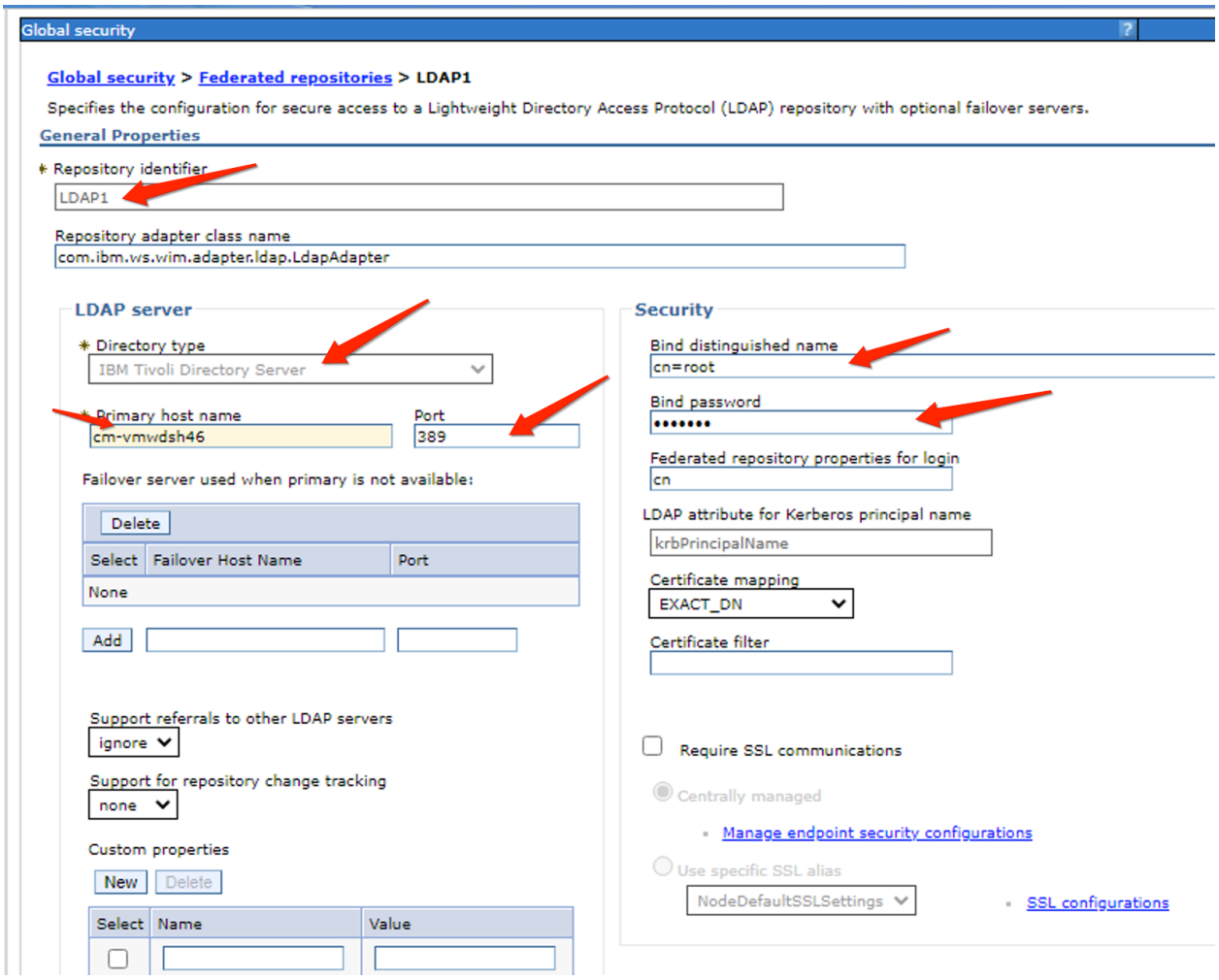


6. On the New page, type a repository identifier, such as **LDAP1** into the Repository identifier field.

7. Specify the LDAP directory that you are using in the **Directory type** field.

8. Type the host name of the primary LDAP directory server in the **Primary host name** field. The host name is either an IP address or a domain name service (DNS) name. Also supply the **port** if needed.

9. Provide values for the **Bind distinguished name** and **Bind password** fields



10. Click **Apply** and then click **Save**.

11. Edit the **Additional properties** to set the user, group, and membership attributes to match to your CPE ldap configuration. Please note that it is very important to get the ldap configuration identical to the CPE tWAS ldap configuration.

12. On the Federated repositories page, use **Manage Repositories** under Related Items to navigate to the newly created repository and verify the configuration matches the one seen in the CPE tWAS.

13. On the respository General Properties page under **Related Items,** click on the link **LDAP Test Query**

14. Make sure to check you have the correct values for **LDAP server Host, Port, Base DN, Bind DN.**

15. Enter the **Bind Password** for the **Bind distinguished Name** user.

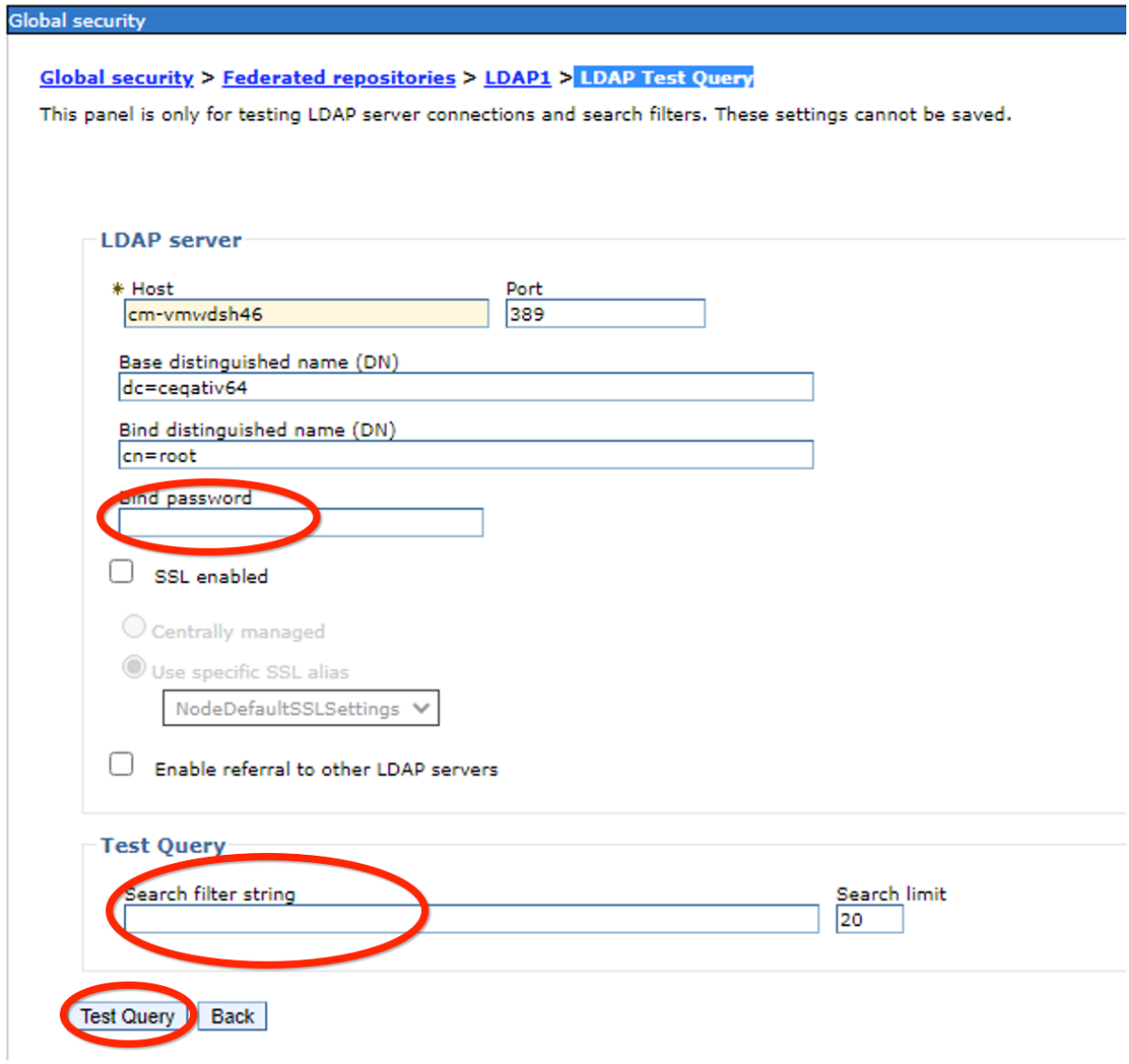
16. Enter the **Search filter string**. For example for IBM SDS utilize a filter that uses the objectclass organizationalPerson to search for all users whose common name (cn) begins with “test”.

    ```
    (&(objectclass=organizationalPerson)(cn=test*))
    ```

    Or leave it as blank to return all entries.

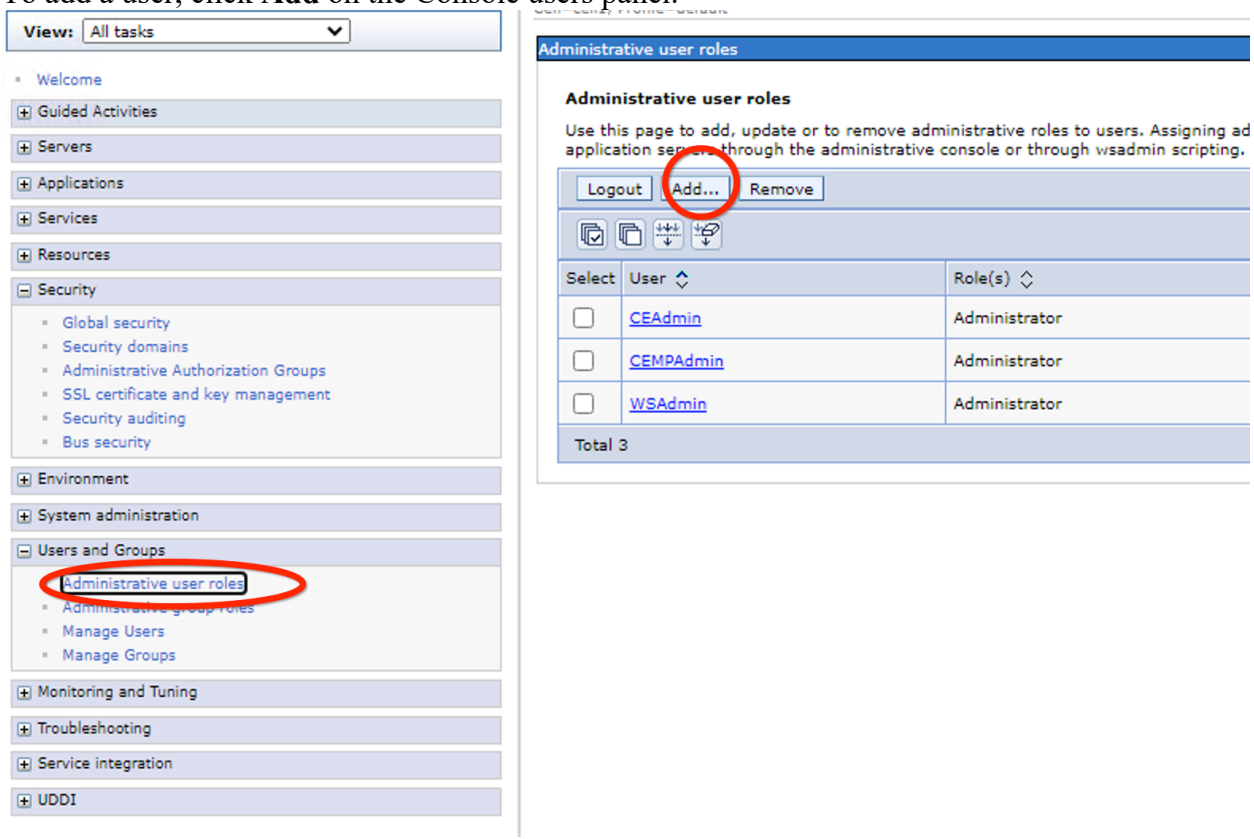
17. Click **Test Query**

18. This results in a page with the LDAP entries, if you get any error adjust the values on the page until you see the ldap entries corresponding to your query



The following procedure is a second method to test the LDAP configuration and also add a particular user the administrative security role within the application server.

1. From Left Navigation, Click **Users and Groups**. Click Administrative **User Roles** .
2. To add a user, click **Add** on the Console users panel.



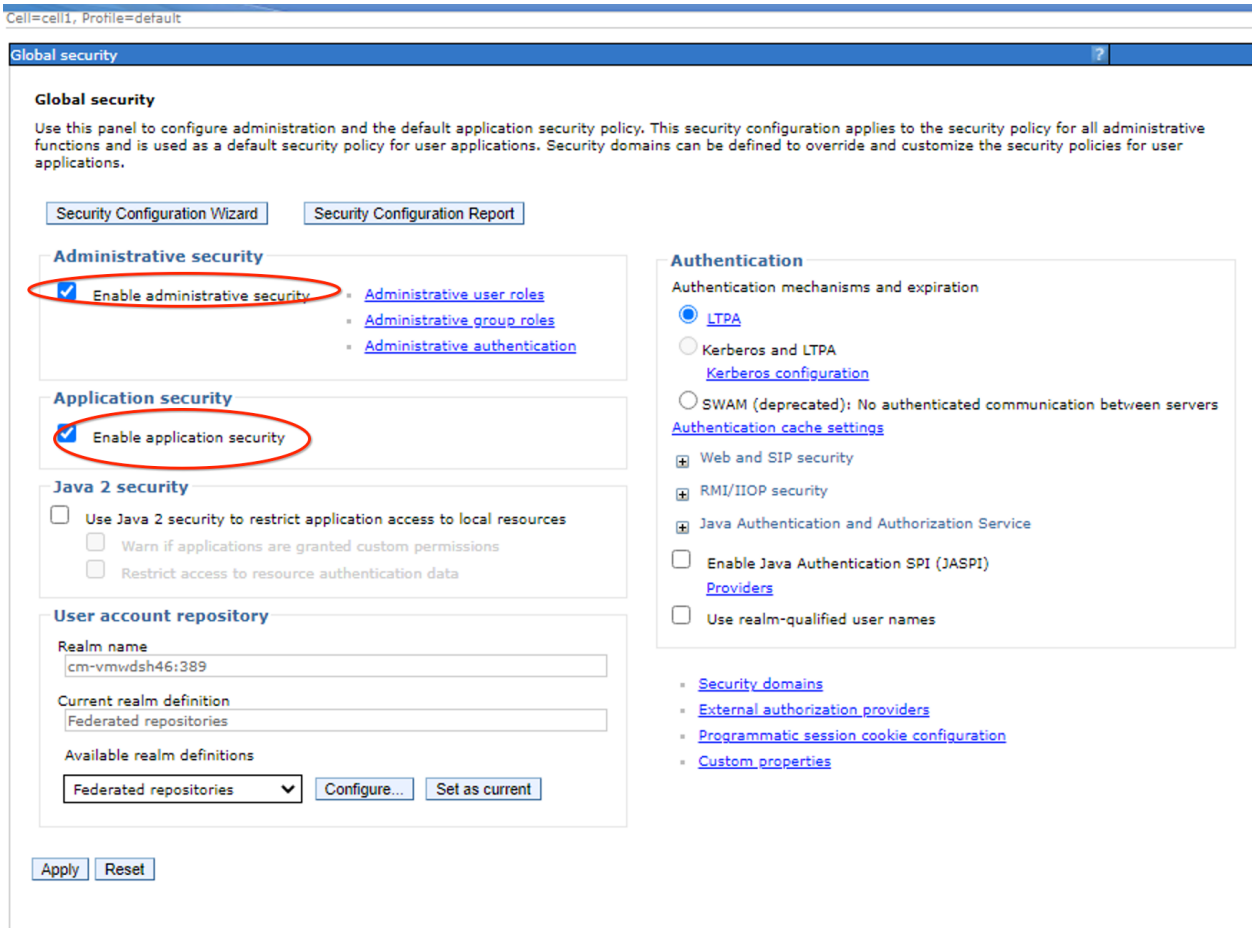
3. To add a new administrator user, follow the instructions on the page to specify a user, and select the **Administrator** role.
4. Enter the appropriate **Search string**. example, user* displays only users with the user prefix. Click **Search**
5. Select the **Available** users and select the right arrow to add to the **Mapped to Role.** If there are no users **Available**, then adjust the **Search string** or fix the ldap configuration.
6. Once the user is added to the Mapped to role list, click **OK**. The specified user is mapped to the security role.
7. After the modifications are complete, click Save to save the mappings.
8. Restart the application server for changes to take effect.

Enable WebSphere Application Security and Administrative security

1. In the WebSphere Application Server administrative console, click **Security** > **Global security**.
2. Select **Enable administrative security**. (optional)
3. Ensure **Enable application security** is selected.

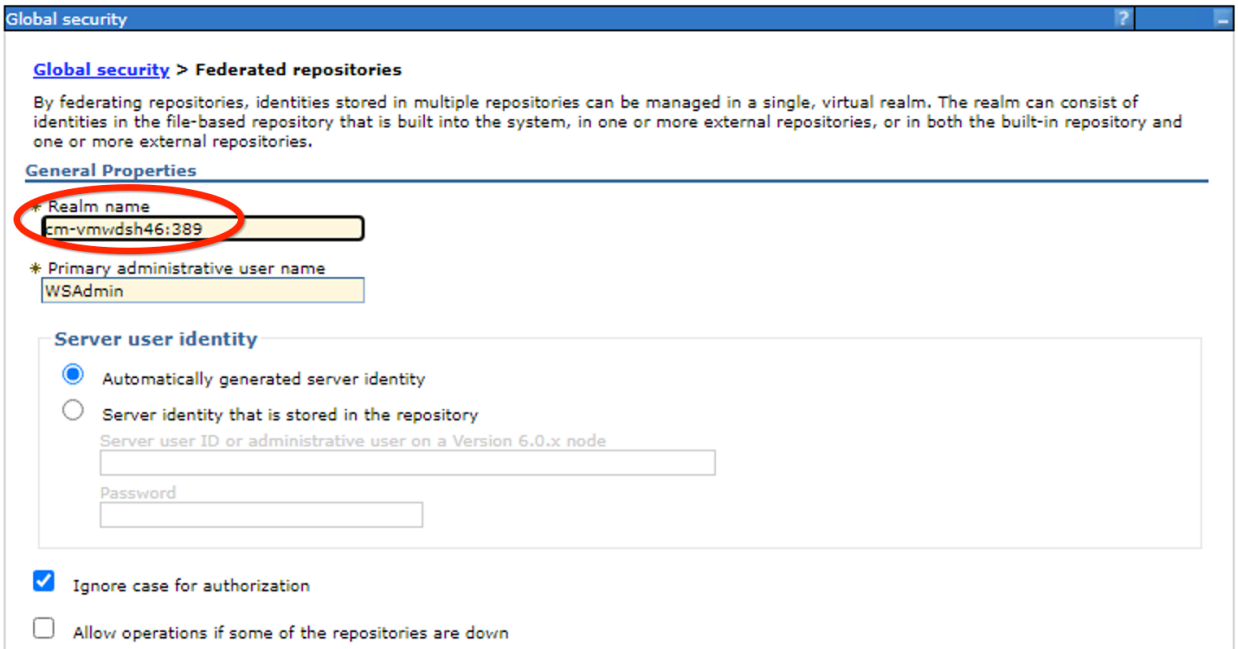
Administrative security protects the WebSphere web console from un authenticated access. Make sure the ldap is properly configured and users are added as the Administrators. If they are any issues with ldap and users, the WebSphere administration console could be inaccessible. To access the administration console and troubleshoot the issues, follow this technote: [Disabling WebSphere administrative security when admin console is not accessible] .

Application Security along with the application's security mapping makes the CS-GraphQL application secured. When you access the CS-GraphQL application this setting configures it to prompt with a Basic Auth dialogue or SSO dialogue depending on the SSO solution chosen.



Realm name in CS-GraphQL tWAS and CPE tWAS

Make sure the Realm name under **Security > Global security > Federated Repositories** matches the realm in the LDAP configuration of your CPE tWAS server.

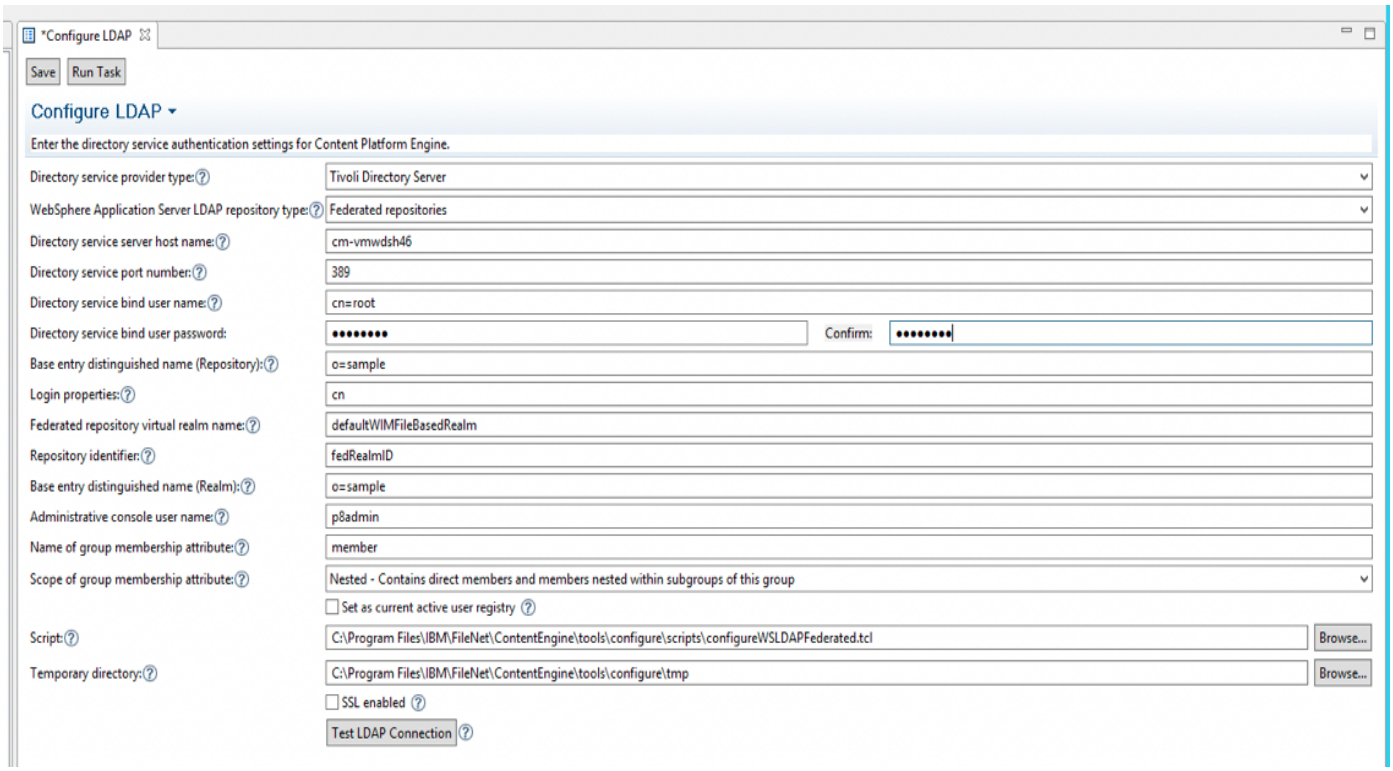


Option 2: Create CSGQL Federated repositories configuration using CPE CMUI profile

If the CPE tWAS LDAP configuration was configured through the configuration tool (configmgr or CMUI) provided with CPE, it is possible to leverage that tool. The CMUI profile utilized previously to create the LDAP configuration for the CPE can be reused against the GraphQL tWAS to configure the ldap using the same configure LDAP task.

1. If the tWAS for CSGQL is located on a different system, use the FileNet Content Manager server installer to install the configuration manager onto the tWAS system where CSGQL will be deployed.
2. Make a copy CPE configuration manager profile to edit and use with CSGQL.
3. Load the copied profile into CMUI which has access to the tWAS for CSGQL.
4. Use the "Edit Application Server Properties" dialog to modify the CMUI profile to point to the CS-GraphQL tWAS. Use "Test Connection" to verify the ability to connect to the tWAS for CSGQL.
5. Run the Configure LDAP task used previously to configure the tWAS for the CPE. This creates the same ldap configuration as in the CPE, including the base DN and realm name. If there are any additional changes that were done in CPE tWAS ldap configuration after the configmgr task was initially rn, apply the same changes to the CSGQL tWAS ldap configuration as well.

Sample values used to complete the configmgr task "Configure LDAP" with the configuration manager user interface (CMUI) for a system using IBM Security Directory Services.



Option 3: Use deployment helper scripts to create CSGQL federated repositories

By running all the scripts using , or by running the subset that target the tasks to configuration the federated repositories and LDAP servers, you can use the automation helper scripts to create this part of the configuration. See the `CSGraphQLAPIDeployScripts/websphere/readme.md` file for more information.

Deploy Content Services GraphQL

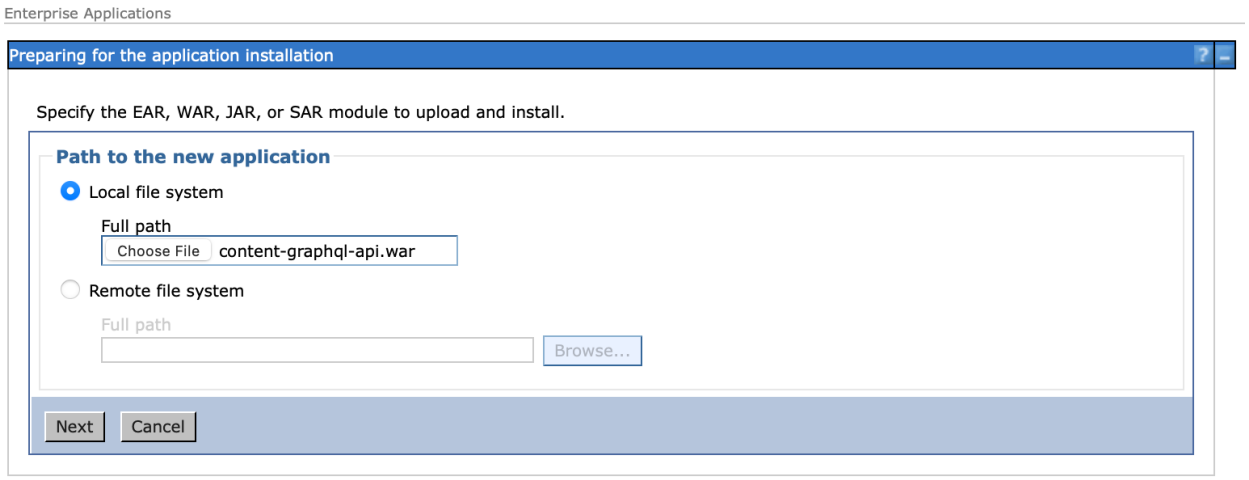
Note: If you choose to use the deployment helper scripts, this entire topic can be skipped. However the information here can be used to verify the deployment performed by the helper scripts is correct or to troubleshoot issues.

Deploy war file

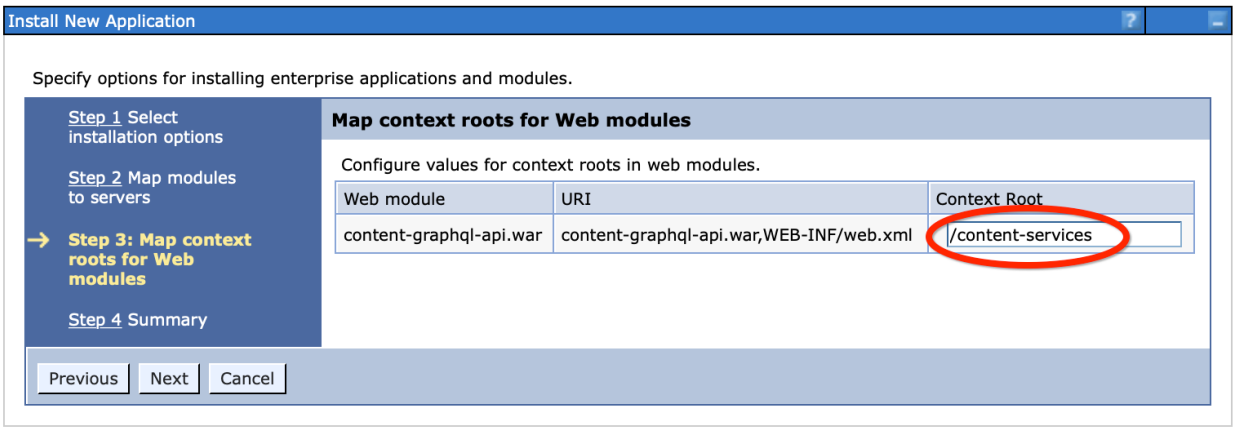
From the CPE media or installation location find the content -graphql-api.war and deploy the application onto GraphQL tWAS.

1. Open the WebSphere Integrated Solutions Console.
2. Click **Applications** > **Application Types** > **WebSphere enterprise applications**.
3. Click **Install**
4. Under **Path to the new application**, select **Remote file system**.
   Choosing **Remote file system** works for both local drives and network drives.

5. Under **Full path** enter the path to the content -graphql-api.war web application file. The content -graphql-api.war file can be found in the lib directory of the CPE Client installed directory



6. Click **Next** to accept all default options until you reach the Map context roots for web modules page.
7. In the Map context roots for web modules, set **Context Root** to /content-services, and click **Next**.



8. Click **Finish**.
9. Verify that the content -graphql-api.war application was installed correctly and click **Save directly to the master configuration**.

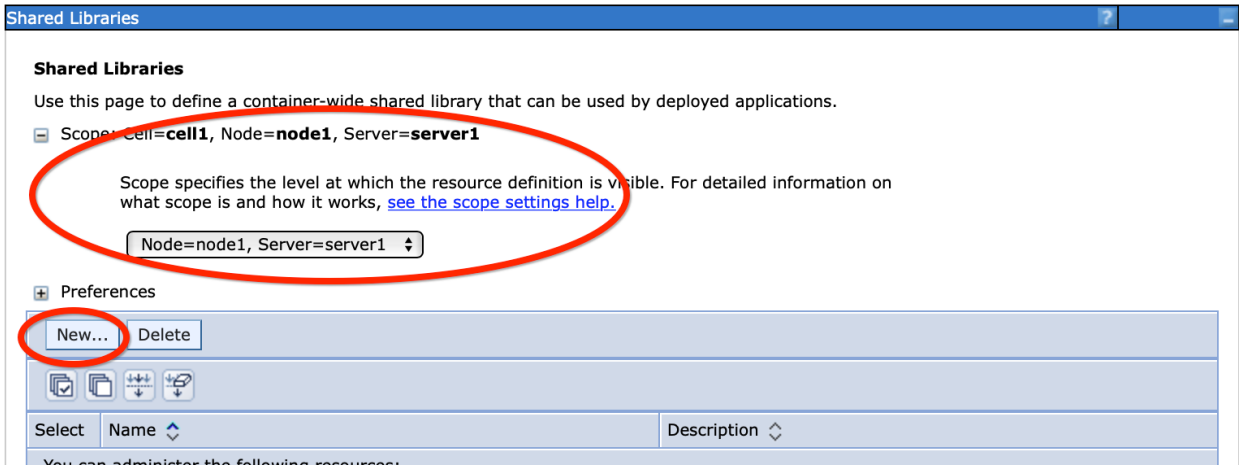
Add and configure required shared libraries

Create shared Libraries

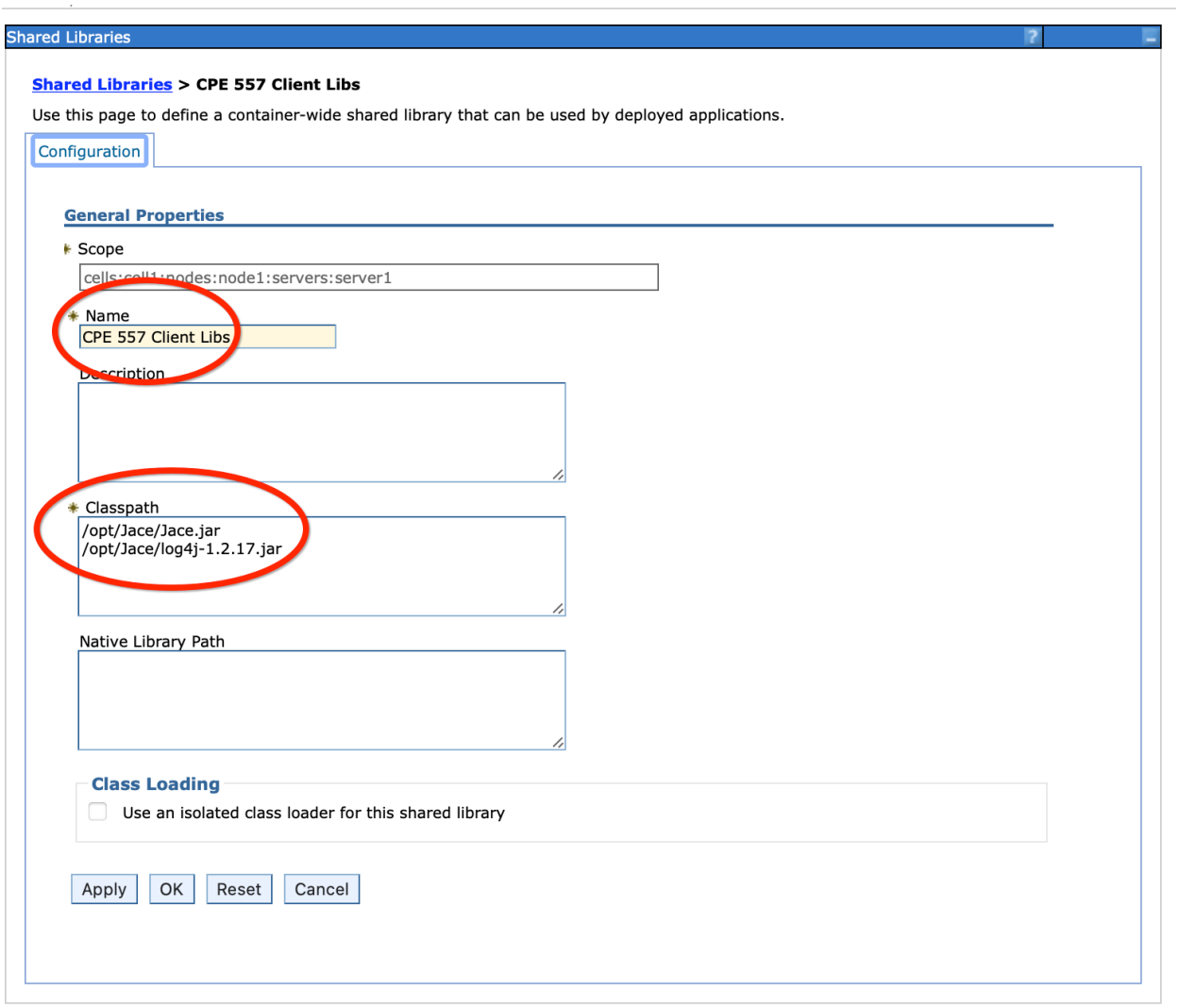
CS GraphQL API app is not built with the CPE client Jar (Jace.jar). So, when an application is being deployed Jace jar has to be loaded as an external library. Some older version of CPE (before 556) also requires log4j jar. This step we are going to create a shared library in CS-GraphQL tWAS and associate the shared library with the content-graphql-api_war application.

The Jace.jar can get from the lib directory of the CPE client installed location. Make sure that the version Jace.jar matches the version of CPE this GraphQL server is connecting to.

1. Expand **Environment** and select **Shared Libraries**.
2. Make sure the scope is set appropriately and create a new shared library.



3. Click **New** and fill out the correct information including the **Name** (CPE 557 Client Libs) and appropriate **Classpath**. Click **Apply**.
4. Make sure these jars files are available to CS-GraphQL tWAS locally at the folder specified and has read permissions

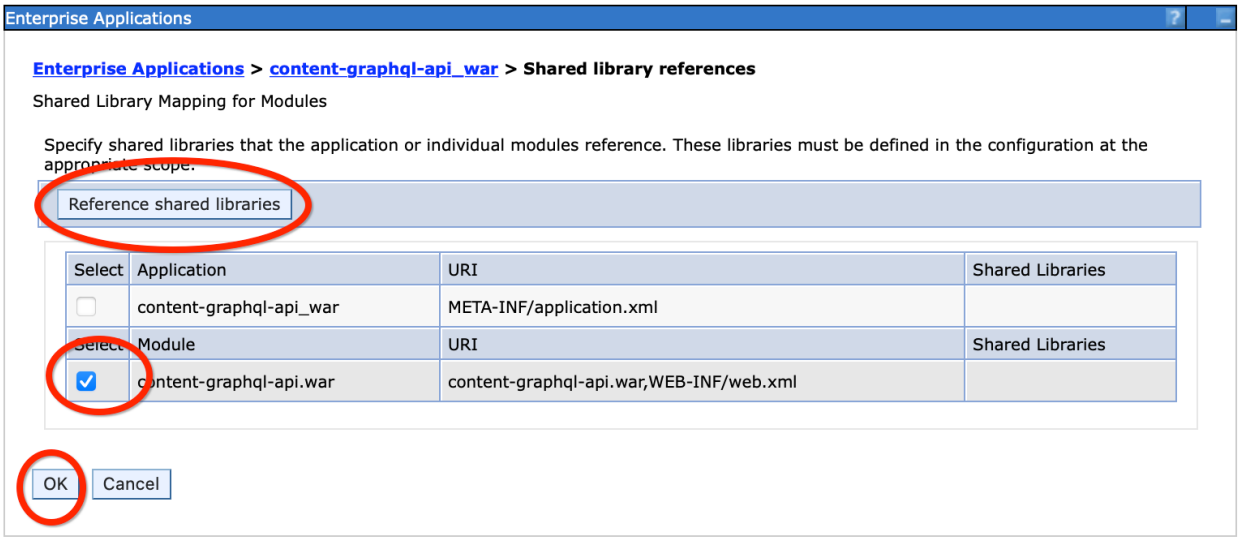


NOTE: Do NOT Select the checkbox "**Use an isolated class loader for this shared library**"

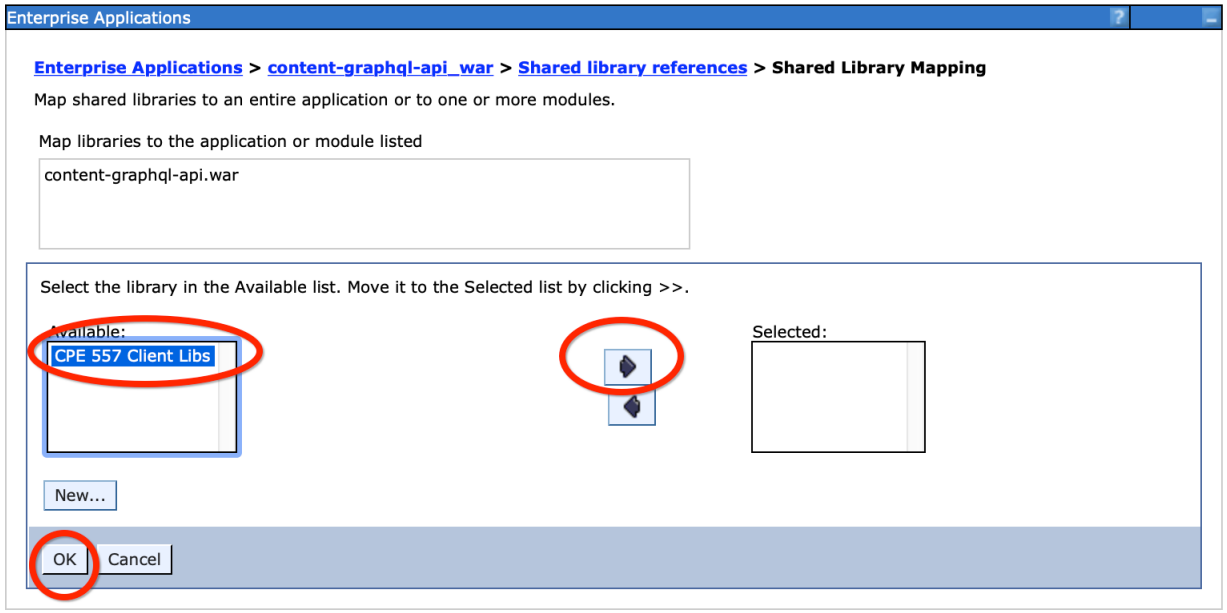
5. Click **OK**, save the configuration

Associating the shared library with the GraphQL application

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications**.
2. Select the **content-graphql-api_war** Application and then select **Shared library references** in the **References**. Select Web module (second checkbox) **content-graphql-api.war**, then click on **Reference shared libraries**.



3. Select the newly created library (CPE 557 Client Libs) and click on the right arrow to add the shared library to **Selected** textbox. Click **OK**.

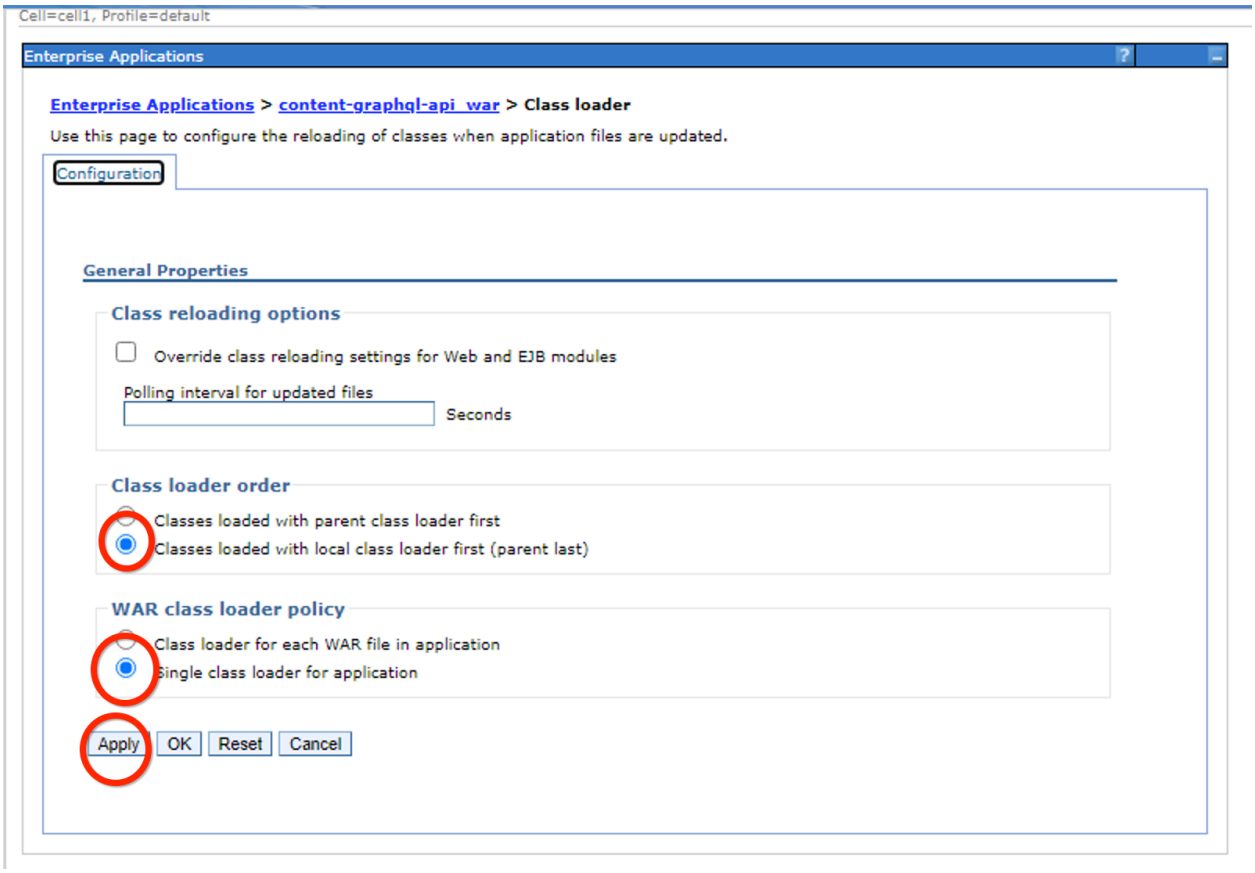


4. The shared library is now associated with the web module. Click **OK**, save the configuration, and then restart the application for the change to take effect.

Set the parent last class loader policy

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications**.
2. Select the **content-graphql-api_war** Application and then select **Class loading and update detection** in the **Detail Properties**.
3. Under **Class Loader order**, select the **Classes loaded with local class loader first (parent last)** radio button.
4. Under **WAR class loader policy**, select the **Single class loader for application** radio button.

5. Click **OK**, save the configuration.



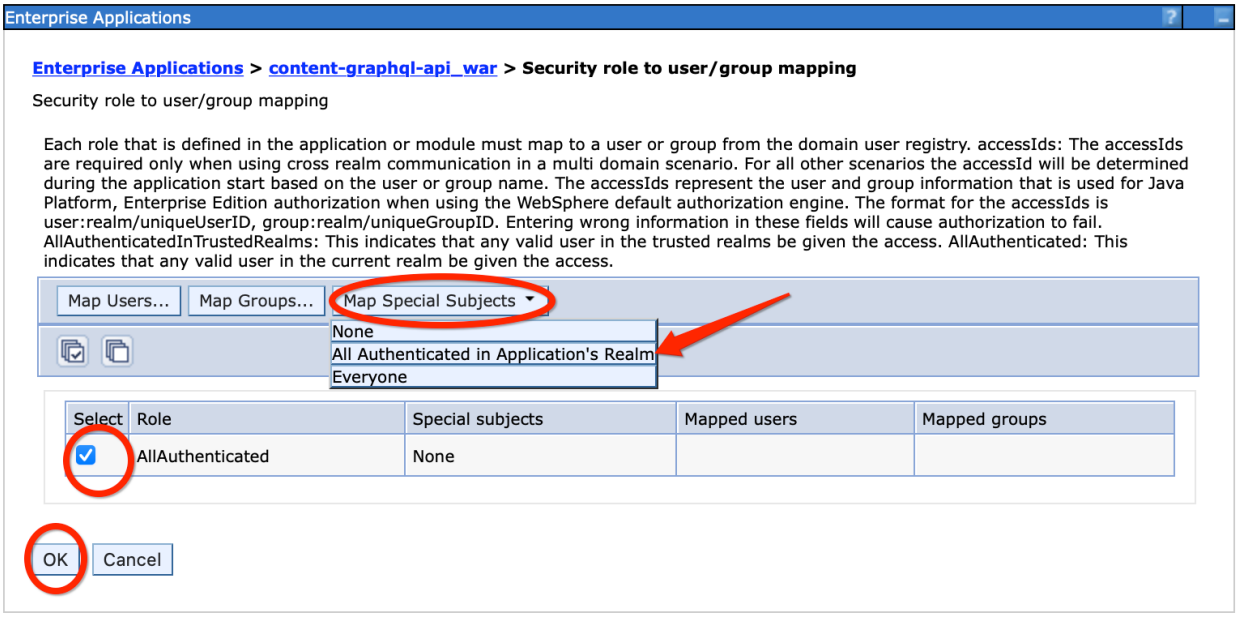
Configure the CS-GraphQL application

Note: If you choose to use the deployment helper scripts, this entire topic can be skipped. However the information here can be used to verify the configuration produced by the helper scripts is correct or to troubleshoot issues.

Map Application Security to all authenticated

In this step, we are associating the All Authenticated in Trusted Realms or All Authenticated Users to the application. So only authenticated users will get the access to the application. Before this step, ldap must have been configured and security must have been enabled.

1. In the administrative console page, click **Applications > Application types > WebSphere enterprise applications**.
2. Select the content-graphql-api_war Application. Under **Detail Properties**, click **Security role to user/group mapping**.
3. Select Role **AllAuthenticated.**



4. Click **Map Special Subjects**, that results in a drop down box.
5. From the drop down select All Authenticated In Trusted Realms or All Authenticated in Application's Relam.
6. Click **OK** and **Save changes to the master configuration**.

Set JVM arguments

1. In the Administration Console select **Servers**
2. Expand **Server Type** and select **WebSphere application servers**
3. Click on the name of your server
4. Expand **Java and Process Management** and select **Process Definition**.
5. Under the **Additional Properties** section, click **Java Virtual Machine.**
6. Scroll down and locate the textbox for **Generic JVM arguments**.
7. Enter the following JVM arguments without comments, .
   Note - tWAS doesn't accept comments as part of JVM arguments Make sure to remove the comments. Comments are shown below to explain the JVM arguments only

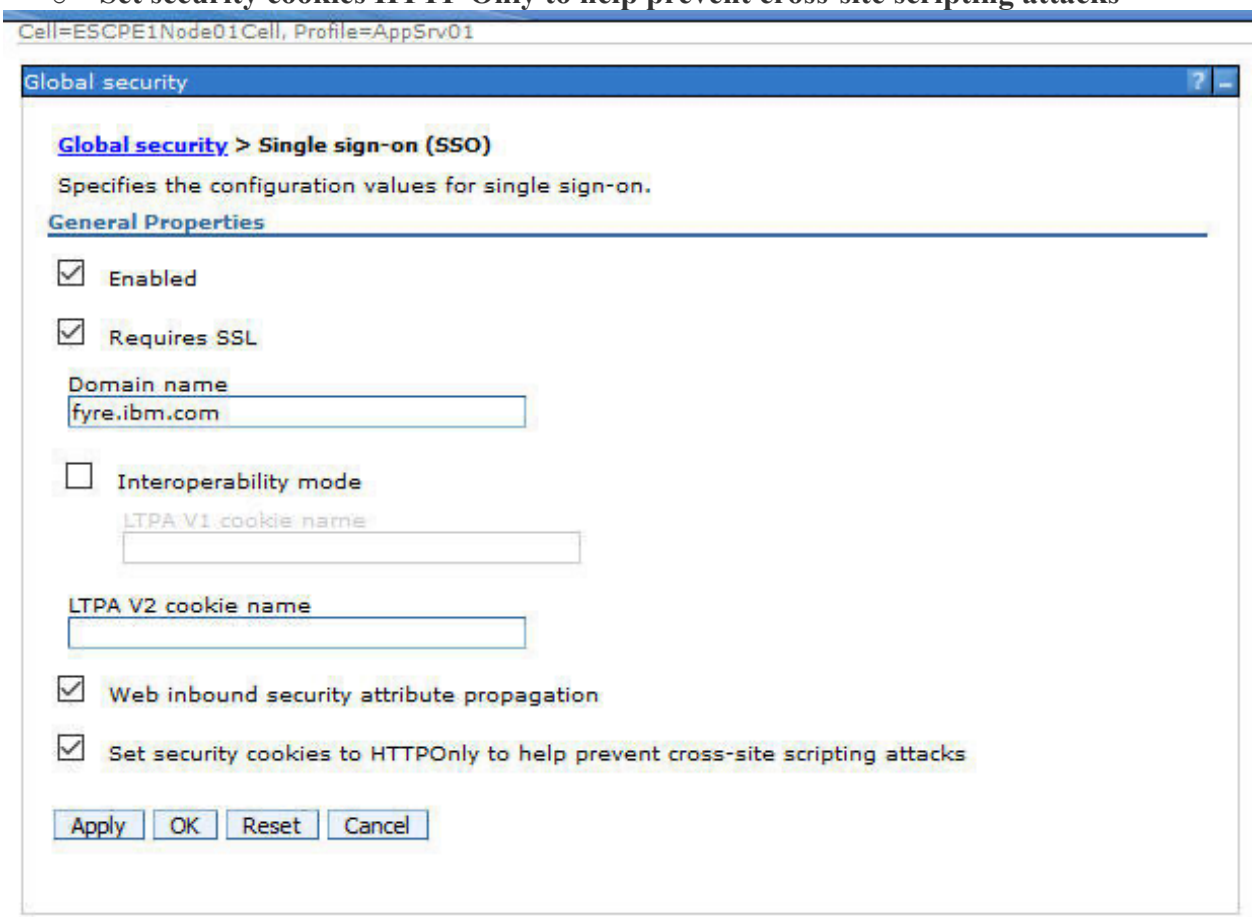
```
#When the CPE Metadata is authored the cached Client Metadata Cache on GraphQL
# becomes stale. This Interval in seconds refreshes the Metadata Cache. Use it only when
there is a metadata authoring at CPE
-Dmetadata.cache.refresh.interval=120
#GraphIQL is GUI client included with application to test the graphql queries. #This
option disables several security checks like CORS, XSRF. Its not #recommended to use in
production
-Dcom.ibm.ecm.content.graphql.enable.graphiql=TRUE
-Dcom.ibm.ws.http.options.writeTimeout=180
-Dcom.ibm.ws.http.options.readTimeout=180
#CPE MTOM URI for this GraphQL server to communicate.
-Decm.content.remote.cpeuri=https://abc.example.com:9443/wsi/FNCEWS40MTOM/
-Dhttps.protocols=TLSv1.2
#WSI Auth token order, for basic auth  use the order ltpa,oauth
# for oauth use the order oauth,ltpa
-Dcom.filenet.authentication.wsi.AuthTokenOrder=oauth,ltpa
# this automatically detects the LTPA or OAuth token in the request
-Dcom.filenet.authentication.wsi.AutoDetectAuthToken=true
-Dsun.net.http.retryPost=false
# allowed origin JVM argument needs to specify
-Decm.content.graphql.cors.enable=true
```

8. Restart the CS-GraphQL tWAS

Enable Single Sign On

Use of single sign on is strongly recommended for production environments.

1. In the WebSphere Application Server administration console, navigate to **Security > Global security > Single sign-on**, enter a Domain Name, and check the following settings:
    - **Enabled**
    - **Requires SSL (with Domain name)**
    - **Web inbound security attribute propagation**
    - **Set security cookies HTTP Only to help prevent cross-site scripting attacks**

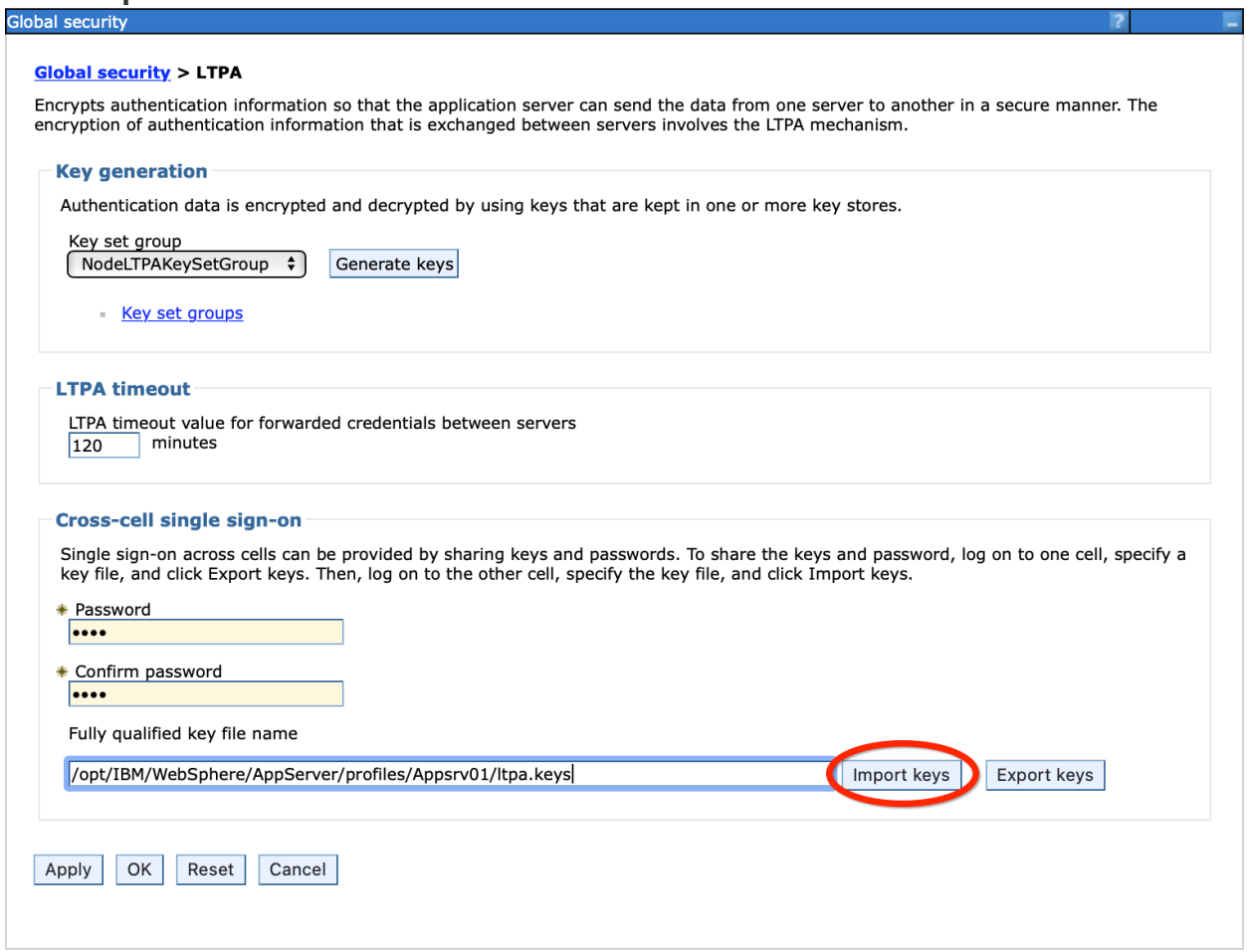


LTPA keys import

Make sure the LDAP realm name is same for the CPE and CSGQL systems. LTPA keys works within the realm. It is important to configure ldap and realm before proceeding so the keys exported from the CPE match the realm the key is imported into for use by the CSGQL.

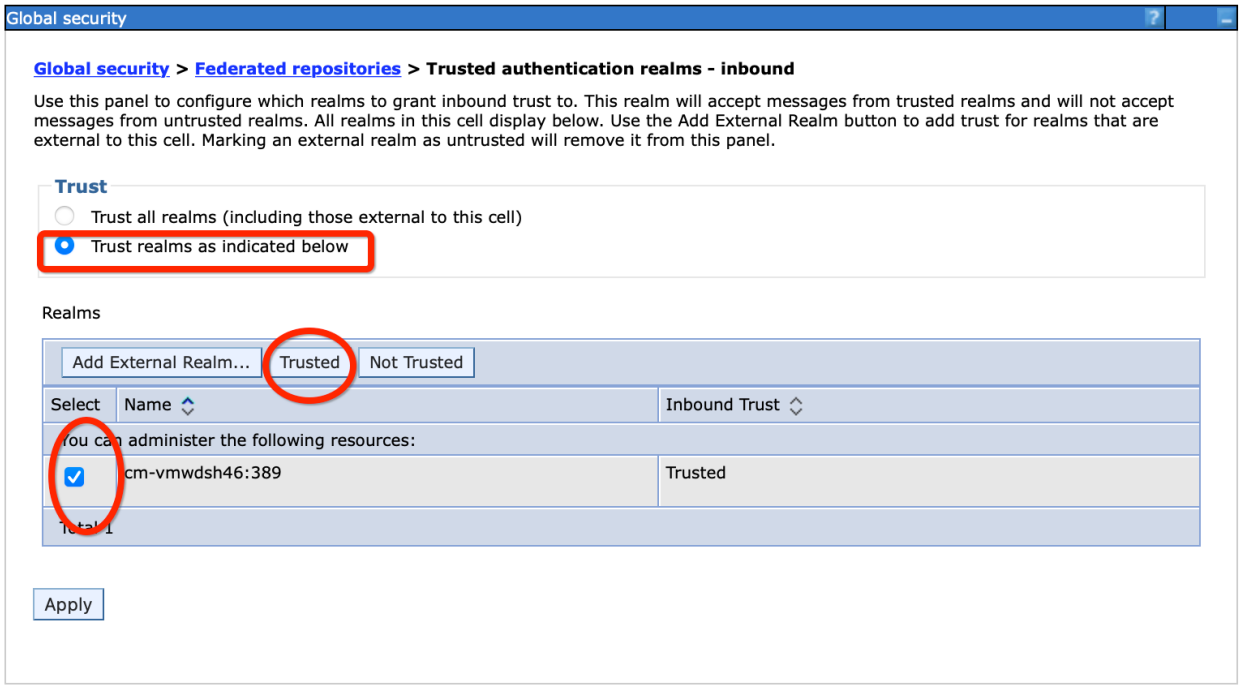
Import LTPA keys

1. Copy the LTPA keys exported on to the GraphQL tWAS system for example, /opt/IBM/WebSphere/AppServer/profiles/Appsrv01/ltpa.keys.
2. In the GraphQL tWAS administration console, navigate to **Security > Global security > LTPA**.
3. Provide a password for the ltpa.keys file. You entered this password while exporting this file
4. Provide a path for the LTPA file you exported from CPE tWAS, for example, /opt/IBM/WebSphere/AppServer/profiles/Appsrv01/ltpa.keys.
5. Click **Import**



Configure inbound trusted realms

1. Click **Security > Global Security**.
2. Select **Federated repositories** click **Configure**
3. Under Related Items, select **Trusted authentication realms - inbound**.
4. Select the **Trust realms as indicated below**
5. Select the check box next to realm name created in previous steps and click **Trusted**



Configure Secure (SSL) communication between CPE and GraphQL

If using a self-signed certificate on the CPE server, and you previously exported the certificate as a part of preparation, import it into the tWAS instance for the CSGQL API application now.

1. Navigate to the profile to host the CSGQL deployment

```
cd <APP_SERVER_ROOT>/profiles/AppSrv01/config/cells/<CELL_NAME>/nodes/<NODE_NAME>
```

2. Copy the `cpe.pem` file previously exported to the CS-GraphQL tWAS server location `<app_server_root>/profiles/AppSrv01/config/cells/<CELL_NAME>/nodes/<NODE_NAME>/cpe.pem`

3. Example command to run the keytool utility to import the certificate

```
<APP_SERVER_ROOT>/java/8.0/bin/keytool -importcert -keystore trust.p12 -storetype
pkcs12 -storepass WebAS -alias cpe -file cpe.pem
```

- Ensure the CSGQL API application server is configured with the JVM argument for CPE URI(-Decm.content.remote.cpeuri) that utilizes the https end point.

Validate the Configuration

With this configuration, the CSGQL API is available using the Basic authentication. To validate the content-graphql-api_war application, access the CsGQL ping page

`http://server:port/content-services/ping`

by supplying the host and port information for the deployed CSGQL application.

This URL produces a result similar to

```
{
"Build-Date" : "May 04, 2021 at 09:21",
"Implementation-Title" : "IBM FileNet Content Services GraphQL API - content-graphql-api",
"Implementation-Version" : "20210504-0921-571-Administrator",
"Product-Version" : "5.5.7",
"Build-Number" : "571"
}
```

If you are having issues accessing the validation ping page, examine the deployment and context root to ensure they match with what appears for the application in WASt. Also review the contents of the tWAS SystemOut.log for the instance hosting the CSGAL application and verify it reports that the content-graphql-api.war file started successfully.

Ping only proves that the application deployed properly. It does not validate the CS-GraphQL API functionality and does not confirm the CPE and CS-GraphQL API are communicating.

To test the CS-GraphQL API functionality access `http://server:port/content-services/` by supplying the host and port information for the deployed CSGQL application. This comes up with the GraphIql user interface if this JVM argument is present:

`-Dcom.ibm.ecm.content.graphql.enable.graphiql=TRUE`

Enter the following GraphQL Query (replace the sample object store name OS1 with the name of an Object Store in your local FileNet P8 domain)

```
{
  _apiInfo(repositoryIdentifier:"OS1")
  {
    buildDate
    implementationVersion
    buildNumber
    productVersion
    implementationTitle
    cpeInfo {
      cpeURL
      cpeUser
      repositoryName
    }
  }
}
```

Debug

If you see `"cpeInfo" : null`, it implies that there is an issue with the connection between CS-GraphQL and CPE.

1. Check the JVM argument with the cpeuri value is correct and specifies the MTOM endpoint for the CPE services

```
-Decm.content.remote.cpeuri=https://myhost.company.com:9443/wsi/FNCEWS40MTOM/
```

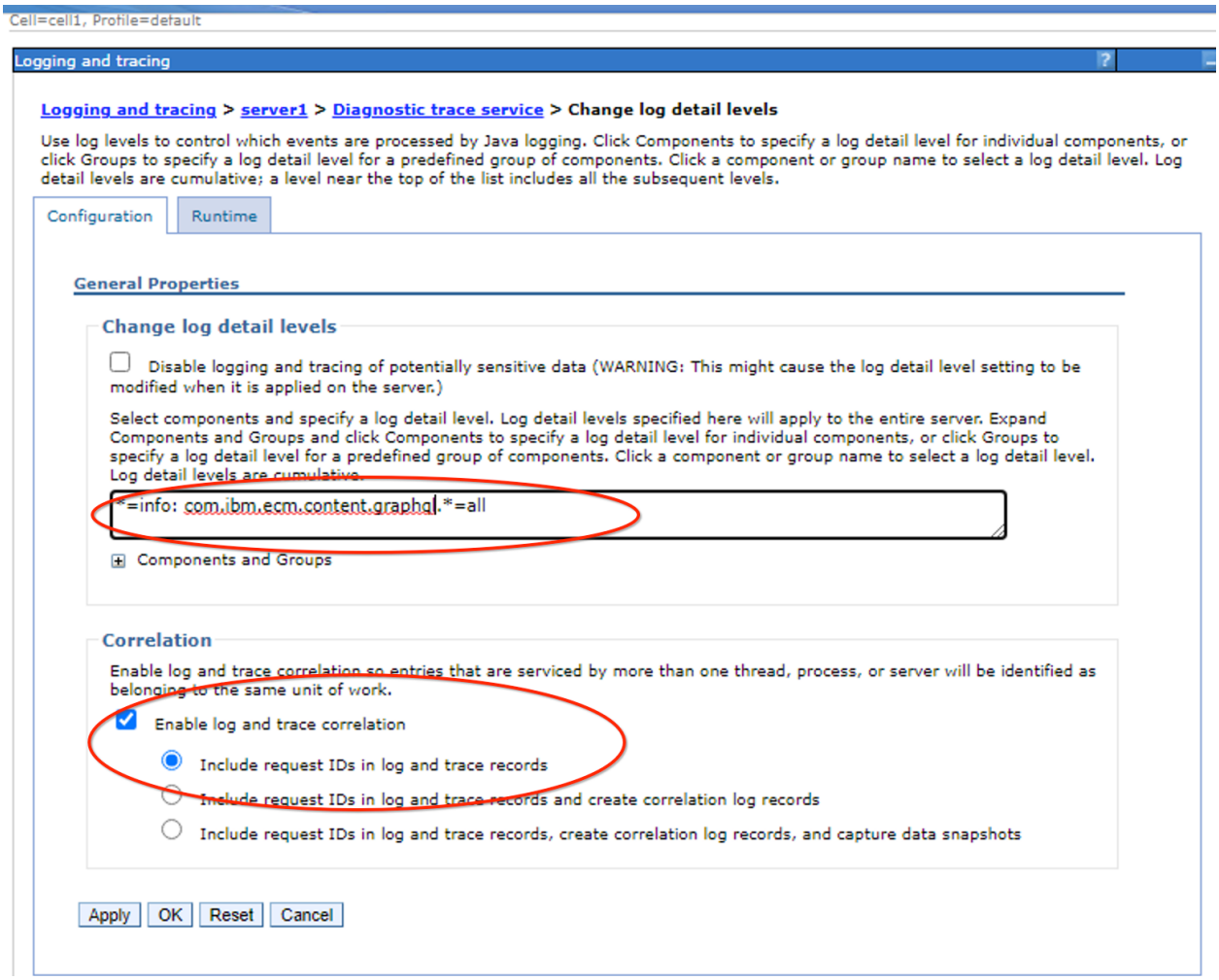
2. Verify the LTPA configuration between the CPE and CS-GraphQL by following the steps in the section above Enable Single Sign On to ensure the configuration is correct.

Trace flags

Set the following trace flags in CS-graphQL tWAS to trace the GraphQL system. Do not enable these flags in production.

Option1: Set trace flags manually using the WASt administrative interface

1. Open the WebSphere Application Service Integrated Solutions Console for the WASt instance hosting the CSGQL application.
2. Expand **Troubleshooting** and select **Logs and trace**.
3. Select the server on which you want to enable traces, and then select **Diagnostic Trace**.
4. Click the **Runtime** tab.
5. Click **Change Log Detail Levels**.
6. Enter `*=info:com.ibm.ecm.content.graphql.*=all` in the Change Log Level Details text box.
7. To trace LTPA token issues add `com.ibm.ws.security.ltpa.LTPAToken2=all`
8. Select **Enable log and trace Correlation** with **Include request IDs in log and trace records**



Option2: Set trace flags using the helper scripts

Trace flags can be set by using the this script will enable or disable logging for GraphQL based on the setting of ENABLE_GQL_DBG and ENABLE_LTPA_DBG in the configureGQL.properties file.

```
debugGQL.sh <properties_file>
```

For information about the script, see the readme.md file located in the `CSGraphQLAPIDeployScripts/websphere` folder or by accessing the copy in GitHub here.

Success

For a successful configuration you will see the response as shown below

GraphiQL ▶ Prettify History

```
1  {
2    _apiInfo(repositoryIdentifier: "FVTTestImportedOS1") {
3      buildDate
4      implementationVersion
5      buildNumber
6      productVersion
7      implementationTitle
8      cpeInfo {
9        cpeURL
10       cpeUser
11       repositoryName
12     }
13   }
14 }
15
```

```
{
  "data": {
    "_apiInfo": {
      "buildDate": "March 17, 2021 at 09:41",
      "implementationVersion": "20210317-0941-530-pwtest330",
      "buildNumber": "530",
      "productVersion": "5.5.7",
      "implementationTitle": "IBM FileNet Content Services GraphQL API - content-graphql-api",
      "cpeInfo": {
        "cpeURL": "https://scow1.fyre.ibm.com:9443/wsi/FNCEWS40MTOM/",
        "cpeUser": "cn=CEAdmin,ou=Shared,ou=Engineering,ou=FileNet,dc=ceqativ64",
        "repositoryName": "FVT Test Imported OS1"
      }
    }
  }
}
```

(Optional) Configure OAuth/OIDC between CS-GraphQL and CPE

Register GraphQL with Identity Provider

All identity providers (IdP) supporting OAuth 2.0 and OpenID Connect authentication have some registration mechanism to identify the client application to the IdP. At a minimum a client id, client secret, and redirect url(s) to the client application are required by the OAuth 2.0 and OpenID Connect specifications.

Note that the same client registration can be used by multiple related applications. So you can register a clientId once and use that same clientId for CPE and its client applications (e.g. ICN, External Share, and Content Services GraphQL).

The following example shows the JSON posted to a UMS registration endpoint to register GraphQL applications. . At present UMS only runs on Liberty. The GraphQL application on tWAS determines the correct format of the redirect URL used in the registration process. A description of each of these parameters can be found in https://www.ibm.com/support/knowledgecenter/SSEQTP_liberty/com.ibm.websphere.wlp.doc/ae/twlp_client_registration.html

```
POST https://my-ums-host:9443/oidc/endpoint/ums/registration
{
    "token_endpoint_auth_method": "client_secret_basic",
    "scope": "openid profile email",
    "grant_types": [
        "authorization_code",
        "client_credentials",
        "implicit",
        "refresh_token",
        "urn:ietf:params:oauth:grant-type:jwt-bearer"
    ],
    "response_types": [
        "code",
        "token",
        "id_token token"
    ],
    "application_type": "web",
    "subject_type": "public",
    "post_logout_redirect_uris": [],
    "preauthorized_scope": "openid profile email",
```

```
        "introspect_tokens": true,
        "trusted_uri_prefixes": [
            "https://my-cpe-server/",
        “https://my-graphql-client/,
        ],
        "resource_ids": [],
        "functional_user_groupIds": [],
        "client_id": "filenetP8Ums",
        "client_secret": "XXXX",
        "client_name": "FileNet P8 UMS",
        "redirect_uris": [
                        "https://my-cpe-server:9443/oidcclient/redirect/FilenetP8Ums",
           "regexp:https://my-graphql-client:!d*/oidcclient/redirect/FileneP8Ums",

        ],
        "allow_regexp_redirects": true
    }
```

A few comments on this example:

- Multiple **redirect_uris** can be specified if there are multiple applications or multiple instances of the same application that will use this client_id.
- If **allow_regexp_redirects** is true, then you can use regular expressions in the redirect_uris. Prefix the URI with "**regexp:**" if specifying a regular expression, as shown in the examples above.
- The middle part of the redirect_uri depends on which OIDC client configuration you are using for your application.
    - If using **openidConnectClient on Liberty**, then use: /oidcclient/redirect/
    - If using the Relying Party Interceptor on traditional Webshpere, then use: /oidcclient/
- The last part of the **redirect_uri** (e.g. FilenetP8Ums) corresponds to the **id** of the **openidConnectClient on Liberty** or **the Relying Party Interception identifier (e.g. provider_1.identifier) on traditional WebSphere** used in your application configuration.
- The **trusted_uri_prefixes** should correspond to those specified in the **redirect_uris**
- If you need to retrieve the current registration for your application, use a GET request with your client_id at the end of the URL
    - e.g. GET https://my-ums-host:9443/oidc/endpoint/ums/registration/filenetP8Ums
- If you need to update the current registration for your application, use a PUT request with your client_id at the end of the URL with body contents similar to your original POST request
    - e.g. PUT https://my-ums-host:9443/oidc/endpoint/ums/registration/filenetP8Ums

For production OAuth/OIDC

Follow the document at https://community.ibm.com/community/user/automation/blogs/roger-bacalzo1/2020/12/17/how-to-configure-sso-between-icn-and-cpe?CommunityKey=2b67f465-a5fe-4a66-ad25-f5e767b607e3&tab=recentcommunityblogsdashboard for configuring OAuth between a CPE client (e,g CS-GraphQL) and CPE. Follow the steps to Configure the CS-GraphQL tWAS in the blog

When using OAuth, change the CS-GraphQL JVM argument for AuthTokenOrder to prefer sending an OAuth token from CS-GraphQL to CPE.

```
-Dcom.filenet.authentication.wsi.AuthTokenOrder=oauth,ltpa
```

XSRF(CSRF)/CORS headers

Cross site request forgery (XSRF/CSRF) is a Cross site request forgery attack on the server.. to mitigate these attacks, the CS-GraphQL application includes the headers.. these headers will be included in every response that goes out from the CS-GraphQL

Cross-Origin Resource Sharing (CORS) is an HTTP-header based mechanism that allows a server to indicate any other origins (domain, scheme, or port) than its own from which a browser should permit loading of resources.

In CS-GraphQL we added the support for all the CORS HTTP response headers through JVM arguments. By default CORS Response headers through JVM options feature is disabled, to enable the feature of setting the CORS response headers through JVM arguments, set the JVM argument *ecm.content.graphql.cors.enable* to true. This JVM arguments sets the response CORS headers to the default values mentioned below. The following table gives the CORS HTTP response headers, their JVM arguments and default values. These values can be overridden by using the JVM arguments based on the needs of the sample application. Following default values should be sufficient for many applications.

HTTP Header	**JVM argument**	default values
Access-Control-Allow-Origin*	ecm.content.graphql.cors.origin.url	it sets the value to HTTP **origin** header
Access-Control-Allow-Methods*	ecm.content.graphql.cors.allow.methods	GET,POST,OPTIONS,PUT,DELETE,HEAD
Access-Control-Allow-Credentials*	ecm.content.graphql.cors.allow.credentials.boolean	true
Access-Control-Allow-Headers*	ecm.content.graphql.cors.allow.headers	Connection, Pragma, Cache-Control, Navigator-Client-Build, XSRFtoken, Origin, User-Agent, Content-Type, Content-Length, Navigator-Client-Identity, Accept-Control-Request-Method, Accept-Control-Request-Headers, Accept, Referer, Accept-Encoding, Accept-Language, DNT, Host, Content-Length, Cache-control, Cookie
Access-Control-Expose-Headers*	ecm.content.graphql.cors.expose.headers	Content-Length, Content_Type, Content-Language, X-Powered-By, Date, Allow, Transfer-Encoding, $WSEP, DNT, Access-Control-Allow-Credentials, Access-Control-Allow-Headers, Access-Control-Allow-Max-Age, Access-Control-Allow-Methods, Access-Control-Allow-Origin, Access-Control-Expose-Headers, Connection, Cache-control, Cookie
Access-Control-Max-Age*	ecm.content.graphql.cors.max.age.seconds	86400

Sample CORS JVM values

```
-Decm.content.graphql.cors.enable=true
-Decm.content.graphql.cors.origin.url=*
-Decm.content.graphql.cors.allow.methods=GET,POST,OPTIONS,PUT
-Decm.content.graphql.cors.allow.credentials.boolean=true
-Decm.content.graphql.cors.allow.headers=Connection,Pragma,Cache-Control,Navigator-Client-Build,XSRFtoken,Origin,User-Agent,Content-Type,Content-Length,Navigator-Client-Identity,Accept-Control-Request-Method,Accept-Control-Request-Headers,Accept,Referer,Accept-Encoding,Accept-Language,DNT,Host,Content-Length,Cache-control,Cookie
-Decm.content.graphql.cors.expose.headers=Content-Length,Content_Type,Content-Language,X-Powered-By,Date,Allow,Transfer-Encoding,$WSEP,DNT,Access-Control-Allow-Credentials,Access-Control-Allow-Headers,Access-Control-Allow-Max-Age,Access-Control-Allow-Methods,Access-Control-Allow-Origin,Access-Control-Expose-Headers,Connection,Cache-control,Cookie
-Decm.content.graphql.cors.max.age.seconds=86400
```