Enabling SAF Security in Liberty Servers

IBM Washington Systems Center

Mitch Johnson – mitchj@us.ibm.com
Washington Systems Center

Table of Contents

Mitch Johnson (mitchj@us.ibm.com)

Security is added to a Liberty server to provide protection for a Liberty server's resources, e.g., features and applications. Using System Authorization Facility (SAF) to ensure that authentication (validating a client's credentials) and authorization (controlling access to resources) requires the coordination of SAF resources and Liberty server's configuration elements. The purpose of this document is to provide the information an administrator needs to understand how SAF security and Liberty work together by providing information and examples on how to define these resources and how these resources relate to a Liberty server's XML configuration elements

***N.B**. SAF is used as a generic term for the local z/OS security product regardless of whether it is IBM's RACF or Broadcom's ACF2 or TopSecret products. In this document RACF commands will be used in the examples but the key is to remember that the concepts covered are the same for ACF2 and TopSecret. It is the implementation details for these products that will be different.*

Liberty and client credentials

The credentials provided by a client will be authenticated, that is, verified with the SAF registry. The credentials may be in the form of a SAF identity and password combination (*basic security)* or a distributed identity derived from the subject's and issuer's distinguished names from the client's personal certificate (*mutual authentication*) or a distributed identity provided in a field in the payload of a token (*trusted third-party token* or more formally known as a *bearer token*). When the latter two are used for authentication, the distributed identity is mapped to a local SAF identity. If the identity's authentication request fails, the client's request is rejected with an HTTP 401 Unauthorized code. *This document will use the term mapped identity regardless of which method of authentication was used.*

Once the client's credentials are authenticated, it is the mapped SAF identity that will be used for subsequent SAF authorization checks to ensure the identity has the authority to access any SAF protected resources.

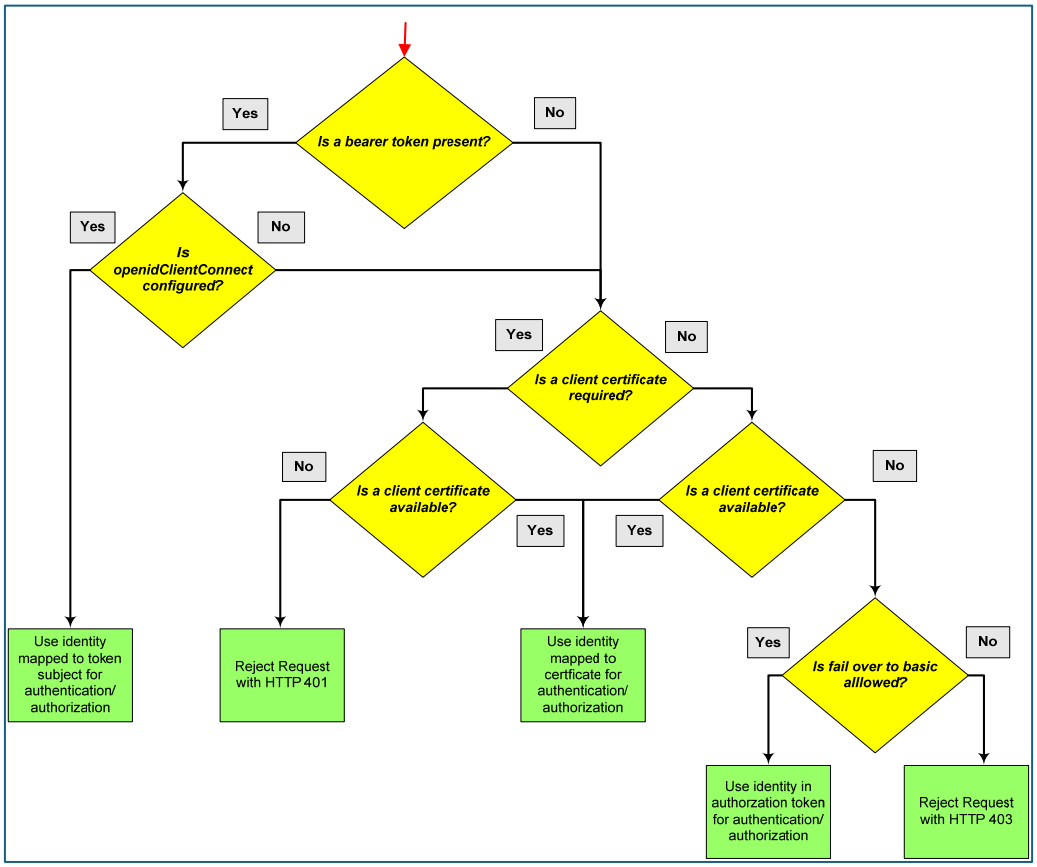
Credential authentication precedence order

Since it is possible for a client's request to include credentials using any of these options, it is useful to understand which has priority.

- If the client request includes a third-party or bearer token and the Liberty server is configured for this token, the distributed identity in the token is mapped to a local SAF identity and this identity is used for authorization checks.
- If the client request does not include a token or if the Liberty server is not configured for the token in the request, the next check is for the presence of information based on a client certificate.
- If the client request does contain certificate information and client authentication support is enabled (clientAuthenticationSupported in the <*ssl*> configuration element) the information provided by the client certificates is mapped to a local SAF identity and this identity is used for authorization checks).
- Finaly if neither bearer token nor client certificate information was provided, the fail-over is to basic security. Note that fail-over to basic security is disabled by default.

Mitch Johnson (mitchj@us.ibm.com)

The flow chart below shows the precedence order when multiple credentials are provided in a connection request.



Liberty and Pass tickets

A past ticket is an alternative for a password for authenticating client credentials with a Liberty server running on z/OS. A pass ticket is generated by using a symmetric key (a value share between the client and server endpoints) to encrypt a combination of an *identity* with an *application name*. Also embedded in the encrypted pass ticket is a time stamp (based on the current Universal Coordinated Time (UCT)). The pass ticket is only valid until a time interval expires (usually 10 minutes after the pass ticket is generated).

Enabling RACF pass tickets for a z/OS application requires two RACF PTKTDATA resources, both with names derived from the value of the *profilePrefix* attribute of the <*safCredentials/*> configuration element, e.g. *BBGZDFLT*.

```
<safRegistry id="saf" />
<safAuthorization racRouteLog="ASIS" />
<safCredentials unauthenticatedUser="WSGUEST" profilePrefix="BBGZDFLT" />
```


Mitch Johnson (mitchj@us.ibm.com)

The first PTKTDATA resource provides the symmetric key (i.e., KEYMASK) to be used to encrypt the pass ticket by both the client and the Liberty server.

```
RDEFINE PTKTDATA applName SSIGNON(KEYMASK(keymaskValue))
APPLDATA('NO REPLAY PROTECTION')
```

Where:

- *applName* is an application name assigned to the resource, e.g., BBGZDFLT
- *keymaskValue* is the value of the secured sign-on application key (symmetric key), a 64-bit hex value
- *replayProtection* indicates if a pass ticket can be reused

The second PTKTDATA resource controls which identities can generate pass tickets, e.g. *IRRPTAUTH.applName.identity.*

```
RDEFINE PTKTDATA IRRPTAUTH.BBGZDFLT.* UACC(NONE)

/*Permit access to PTKTDATA to the Liberty server for all identities for outbound authentication
PERMIT IRRPTAUTH.BBGZDFLT.* ID(LIBSERV) CLASS(PTKTDATA) ACCESS(UPDATE)

/*Permit access to PTKTDATA to a z/OS Client running under SAF identity USER1 to the Liberty server
PERMIT IRRPTAUTH.BBGZDFLT.USER1  ID(USER1) CLASS(PTKTDATA) ACCESS(UPDATE)
```

The unauthenticatedUser

On z/OS, processes or threads must always be running under the authority of a SAF identity. When a client's request arrives at a Liberty server, the client's mapped identity has not yet been determined. So, during this initial phase, the thread is running under the authority of the SAF identity provided by the *unauthenticatedUser* attribute in the server's <*safCredentials*/> configuration element.

```
<safRegistry id="saf" />
<safCredentials unauthenticatedUser="WSGUEST" />
```

- <safRegistry/> enables the use of a SAF registry for authenticating credentials.
- <safCredentials/> identifies which SAF identity should be used for the *unauthenticated user*

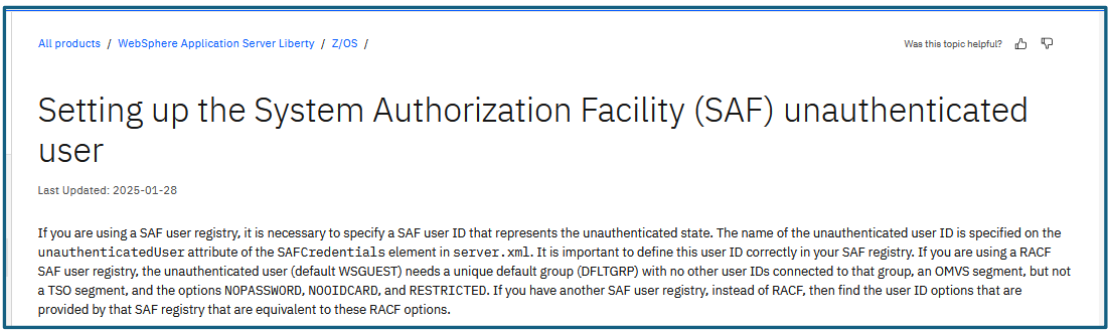
It is only when the client's credentials are successfully authenticated and the mapping performed does the SAF security context of the thread switch to that of the mapped SAF identity. Below is an example of the RACF commands for defining an *unauthenticated user* to RACF with limited authority.

```
ADDGROUP WSGUESTG OMVS(AUTOGID) OWNER(SYS1)

ADDUSER WSGUEST DFLTGRP(WSGUESTG) OMVS(AUTOUID -
  HOME(/u/wsguest) PROGRAM(/bin/sh)) NAME('UNAUTHENTICATED USER') -
  NOPASSWORD NOOIDCARD RESTRICTED
```


Mitch Johnson (mitchj@us.ibm.com)

For more information regarding the authenticated identity, see URL https://www.ibm.com/docs/en/was-liberty/zos?topic=ezaslz-setting-up-system-authorization-facility-saf-unauthenticated-user, see below.



All products / WebSphere Application Server Liberty / z/OS /

Was this topic helpful?

Setting up the System Authorization Facility (SAF) unauthenticated user

Last Updated: 2025-01-28

If you are using a SAF user registry, it is necessary to specify a SAF user ID that represents the unauthenticated state. The name of the unauthenticated user ID is specified on the `unauthenticatedUser` attribute of the SAF`Credentials` element in `server.xml`. It is important to define this user ID correctly in your SAF registry. If you are using a RACF SAF user registry, the unauthenticated user (default WSGUEST) needs a unique default group (DFLTGRP) with no other user IDs connected to that group, an OMVS segment, but not a TSO segment, and the options NOPASSWORD, NOOIDCARD, and RESTRICTED. If you have another SAF user registry, instead of RACF, then find the user ID options that are provided by that SAF registry that are equivalent to these RACF options.

Liberty and z/OS privileged functions

For a Liberty server to be able to use SAF services, the server's SAF identity, i.e., the identity under which the Liberty server is executing, must have access to z/OS privileged functions that are not available to normal identities. Providing access to z/OS privileged functions requires:

- Permitting READ access to the **SERVER** resource profiles protecting these z/OS privileged functions to the server's identity
- But simply having access to these **SERVER** resources is not sufficient by itself. The server's identity must also be authorized to register or connect with an *active* Angel task at server startup. Access to an active Angel task requires READ access to the **APPL** resource profile associated with the Angel.

This section explains how this access is obtained by providing access to SAF **SERVER** and **APPL** resources.

Permitting access to z/OS privileged functions

There are multiple resources related to providing access to z/OS privileges but this document focuses on the resources specific for SAF authentication and authorization.

Access to the authorized server module *BBGZSAFM* and access to the SAF registry and authorization services are required to perform z/OS privilege functions related to authentication and authorization. Access is provided by having READ access to two **SERVER** resources:

- READ access to **SERVER** resource *BBG.AUTHMOD.BBGZSAFM* provides general access to be able to list z/OS Authorized services available for this server.
- READ access to **SERVER** resource *BBG.AUTHMOD.BBGZSAFM.SAFCRED* provides access to be able to use of the SAF authorized user registry and access to the SAF authorization services.

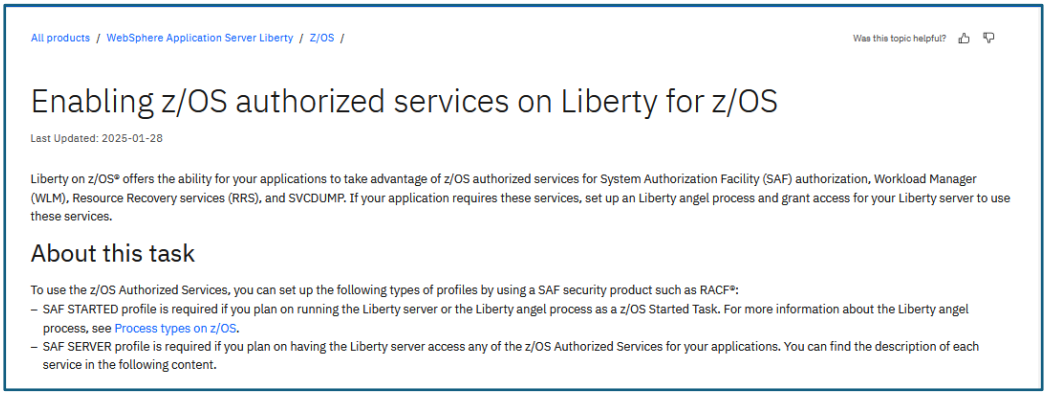
Mitch Johnson (mitchj@us.ibm.com)

Below are examples of the RACF commands that will define and grant access to these resources to members of SAF group LIBGRP (the server identity is a member of this group).

```
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM  CLASS(SERVER) ACCESS(READ) ID(LIBGRP)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.SAFFCRED UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.SAFCRED CLASS(SERVER) ACCESS(READ) ID(LIBGRP)
```

An example of defining all the **SERVER** resources for accessing the other z/OS privileged functions are provided in section *SERVER resources related to accessing z/OS privileges* on page 27. For more information about enabling these and the other z/OS privileged services for a Liberty server see URL https://www.ibm.com/docs/en/was-liberty/zos?topic=liberty-enabling-zos-authorized-services-zos



All products / WebSphere Application Server Liberty / z/OS /

Was this topic helpful?

Enabling z/OS authorized services on Liberty for z/OS

Last Updated: 2025-01-28

Liberty on z/OS® offers the ability for your applications to take advantage of z/OS authorized services for System Authorization Facility (SAF) authorization, Workload Manager (WLM), Resource Recovery services (RRS), and SVCDUMP. If your application requires these services, set up an Liberty angel process and grant access for your Liberty server to use these services.

About this task

To use the z/OS Authorized Services, you can set up the following types of profiles by using a SAF security product such as RACF®:

- SAF STARTED profile is required if you plan on running the Liberty server or the Liberty angel process as a z/OS Started Task. For more information about the Liberty angel process, see Process types on z/OS.
- SAF SERVER profile is required if you plan on having the Liberty server access any of the z/OS Authorized Services for your applications. You can find the description of each service in the following content.

Registering with an active Liberty Angel process

The next consideration is that the identity under which the server is running is allowed to register or connect to an active Angel process. Authorization to access or register with an Angel requires that the server's identity has READ access to the **SERVER** resource profile protecting the Angel. This name of this resource profile is either *BBG.ANGEL* for the default (or no-name angel) or to *BBG.ANGEL.<angelName>* where the <angelName> value matches the *NAME* parameter in the Angel's startup JCL.

Below are examples of RACF command to define the resources that define the required RACF resources and grant access of members of group *LIBGRP* to the default (no name) Angel and an Angel named BBGZDFLT.

```
/* Define a SERVER profile for the default or no name Angel
RDEFINE SERVER BBG.ANGEL UACC(NONE) OWNER(SYS1)
PERMIT  BBG.ANGEL CLASS(SERVER) ACCESS(READ) ID(LIBGRP)

/* Define a SERVER profile for an Angel named BBGZDFLT
RDEFINE SERVER BBG.ANGEL.BBGZDFLT UACC(NONE) OWNER(SYS1)
PERMIT  BBG.ANGEL.BBGZDFLT  CLASS(SERVER) ACCESS(READ) ID(LIBGRP)
```


Mitch Johnson (mitchj@us.ibm.com)

Below is an example of a JCL procedure for the Angel named BBGZDFLT.

```
//BBGZANGL PROC PARMS='',COLD=N,NAME='BBGZDFLT',SAFLOG=Y
//*-------------------------------------------------------------------
//  SET ROOT='/usr/lpp/WebSphere/Liberty'
//*-------------------------------------------------------------------
//* Start the Liberty angel process
//*-------------------------------------------------------------------
//* This proc may be overwritten by fixpacks or iFixes.
//* You must copy to another location before customizing.
//*-------------------------------------------------------------------
//STEP1   EXEC PGM=BPXBATA2,REGION=0M,TIME=NOLIMIT,
//      PARM='PGM &ROOT./lib/native/zos/s390x/bbgzangl COLD=&COLD NAME=X
//             &NAME &PARMS SAFLOG=&SAFLOG'
```

***N.B**. Set the SAFLOG parameter in the Angel's startup JCL to **Y** so that the Angel will log SAF messages related to Angel registration to the SYSLOG during a server's startup.*

The Angel to which a Liberty server is to register is identified using the Angel name in the Java directive *com.ibm.ws.zos.core.angelName.* For example, to register with an Angel named BBGZDFLT, use the Java directive below:

-Dcom.ibm.ws.zos.core.angelName=BBGZDFLT

This directive could be provided in *STDENV* input, in the *bootstrap.properties* or in any other method for providing Java directives to a Java Virtual Machine on z/OS. If this Java directive is absence, then registration with the default Angel (*BBGZDFLT*) is attempted.

The Liberty Security Domain Considerations

Giving a Liberty server's SAF identity access to perform z/OS privilege functions against SAF resources does require that these privileges be restricted to a subset of SAF resources (e.g., identities, EJBRoles, etc.). This subset of SAF resources is referred to as the server's security domain or realm. This restriction allows an administrator to isolate or separate SAF resources between different sets of Liberty servers, i.e. development, versus test, versus production, etc. This means that the server can only authenticate identities that reside in the server's domain and the server can only perform authorization checks on SAF resources in the server's domain. Note that in Liberty documentation this security domain will be referred to as the WLP z/OS Security Access Domain (WZSSAD).

The server's security domain name is determined by the value of *profilePrefix* attribute of the server's <*safCredentials*/> configuration element.

```
<safCredentials unauthenticatedUser="WSGUEST"
  profilePrefix="BBGZDFLT" />
```

- <safCredentials/> the profilePrefix provides the name of the server's security domain or realm.

Let's see how this name is used to secure the domain's resources.

Mitch Johnson (mitchj@us.ibm.com)

APPL SAF resource

When a client accesses a Liberty server, a SAF check is made for READ access to the SAF **APPL** resource profile protecting the server. This check for READ access is how the server determines if the identity is in the server's domain. The name of the **APPL** resource profile used for this check is determined by the value of *profilePrefix* attribute of the server's <*safCredentials*/> configuration element.

Below are examples of the commands for defining the RACF **APPL** resource and granting access to this resource to the *unauthenticated user*.

```
RDEFINE APPL BBGZDFLT UACC(NONE) OWNER(SYS1)
PERMIT BBGZDFLT CLASS(APPL) ACCESS(READ) ID(WSGUEST)
```

Even the unauthenticated user needs READ access to the **APPL** resource profile protecting the server. In fact, if the unauthenticated user does not have access to the **APPL** SAF resource profile, the message below will appear in the *messages.log* file with the first client authentication request is made.

CWWKS2960W: Cannot create the default credential for SAF authorization of unauthenticated users. All authorization checks for unauthenticated users will fail. The default credential could not be created due to the following error:CWWKS2907E: SAF Service IRRSIA00_CREATE did not succeed because user WSGUEST has insufficient authority to access APPL-ID BBGZDFLT. SAF return code 0x00000008. RACF return code 0x00000008. RACF reason code 0x00000020.

During the authentication process when the mapped identity has been determined, the access of the mapped identity to the APPL resource is also checked. If the mapped identity has READ access to the **APPL** resource profile, subsequent authorization checks are performed using the mapped identity, otherwise, further SAF authorization checks will continue to use the unauthenticated user identity. *This is why it is important that the unauthenticated user has no SAF authority other than READ access to the **APPL** resource profile.*

Removing an identity's APPL Access

Once an identity has been authenticated, SAF authorization information is cached locally in the server's memory. This information will remain cached in the memory until the period of time specified by the *timeout* attribute of the <*authCache*> configuraton element expires, see section *The* Liberty Authentication Cache <authCache> on page 16.

Until the timeout occurs, if an identity's READ access to the **APPL** profile is deleted or removed with a **PERMIT DELETE** command and a ***SETROPTS RACLIST(APPL) REFRESH*** command, the identity does not immediately lose access to the **APPL** resource with the next attempt. The same is true if group access checking is being used and the identity is removed from the group that has READ access to the **APPL** resource. The reason is that the SAF access information for this identity remains cached in the server's memory. For an explanation as to why removing an identity from a group is not sufficient to remove the access, see the information at *Changing the group* membership of an identity on page 16.

Mitch Johnson (mitchj@us.ibm.com)

A MVS MODIFY (F) command can be used to "clear" the server's cached information. Once this command is invoked on the server, the client's identity no longer has access to the **APPL** resource and no longer has access to the server. The MVS MODIFY command to do this for server *BAQSTRT* would be **F BAQSTRT,CACHE,CLEAR,AUTH**.
Liberty does provide a management bean (MBean), see section *18* on page 18 for information about this MBean and examples of how to execute it using cURL, Postman and the API Explorer.

***N.B** Invoking this command does clear all the entries from the cache which means all subsequent will experience the overhead of making SAF service call for the next request.*

SERVER SAF resource

Another **SERVER** resource profile controls access to the security domain. It is being introduced now because the third part of the name of this resource profile must be the same value as the value of the *profilePrefix* attribute used for the **APPL** resource.

This configuration attribute determines the prefix of the SAF resource profile names, (e.g., EJBRoles) on which authorization checks are performed. This prefix is prepended to any application SAF resource when a SAF check is made for authorization. See section *EJBRole SAF checks* on page 9 for information on roles and SAF resource names.

N.B. The prepending of the prefix provides for the ability to define a static application role name to SAF with a different SAF resource names for development execution versus production execution.

Below are examples of the RACF commands that define this **SERVER** resources using the value of the *profilePrefix* attribute, e.g., *BBGZDFLT*. Access to this resource gives the server access to resources in the *BBGZDFLT* domain.

```
RDEFINE SERVER BBG.SECPFX.BBGZDFLT UACC(NONE)
PERMIT  BBG.SECPFX.BBGZDFLT  CLASS(SERVER) ACCESS(READ) ID(LIBGRP)
```

For more details on the requirements for authenticating a user, see URL https://www.ibm.com/docs/en/was-liberty/zos?topic=zos-accessing-security-resources-using-wzssad

EJBRole SAF checks

Each Liberty server resource (e.g. a feature or a function, an application, etc.) is protected by a SAF **EJBRole** resource profile (there are some exceptions for this check for those resources that do not require security).

If the mapped identity does not have READ access to the **EJBRole** profile protecting the resource, the resource request is rejected with an HTTP 403 Forbidden code. The EJBRole profile name used for the SAF check is determined by the Liberty XML configuration element <*safRoleMapper/*>. The default **EJBRole** resource profile pattern for the <*safRoleMapper/*> configuration element is *%profilePrefix%.%resource%.%role%.*

Mitch Johnson (mitchj@us.ibm.com)

- Where *%profilePrefix%* is the value for the *profilePrefix* attribute of the server's *<safCredentials/>* configuration element.
- And where the *%resource%* name of the **EJBRole** resource is the value of the *name* attribute of the <application/>, *<enterpriseApplication/>* or *<webApplication/>* XML configuration element used to define the application. ***N.B.*** *most if not all Liberty administrative features, the %resource% name will be com.ibm.ws.management.security.resources.*
- Finally, the value for *%role%* is determined by the application or feature. For most applications, the value for *%role%* is determined in the *web.xml* file embedded in the application's EAR or WAR files. ***N.B.*** *most if not all Liberty features are protected by the Administrator, Reader and allAuthenticatedUsers roles.*

For a z/OS Connect OpenAPI3 server, an EJBRole SAF check is made for each method of each API, the **EJBRole** default profile name is the concatenation of the *profilePrefix*, the *name* attribute of the *webApplication* XML used to define the APIs WAR file roles as defined the *openapi.yaml* file. For an z/OS Connect OpenAPI3 API requester application, a SAF check is made for only the *invoke* method. As an example, if an z/OS Connect API was deployed using the *<webApplication/>* configuration element below in this server, the **EJBRole** access to invoke a method protected by the *Staff* role would have a name of *BBGZDFLT.cicsAPI.Staff.*

```
<webApplication id="cscvinc" contextRoot="/cscvinc"  name="cicsAPI"
  location="${server.config.dir}apps/cscvinc.war"/>
```

For a z/OS Connect OpenAPI2 server, one **EJBRole**, *zos.connect.access.roles.zosConnectAccess,* is used to protect access for all z/OS Connect OpenAPI2 APIs, Services and API requester applications. That **EJBRole** profile is *profilePrefix.zos.connect.access.roles.zosConnectAccess,* where *profilePrefix* is the only value that varies and is the value of the *profilePrefix* attribute of the *safCredentials* XML. The values for *resource* and *role* are these static strings.

An example for determining the SAF EJBRole resource names

Determining the what EJBRole names that need to be defined to the SAF product may require some investigation. This section shows one way that this can be done using the MQ Console application as an example.

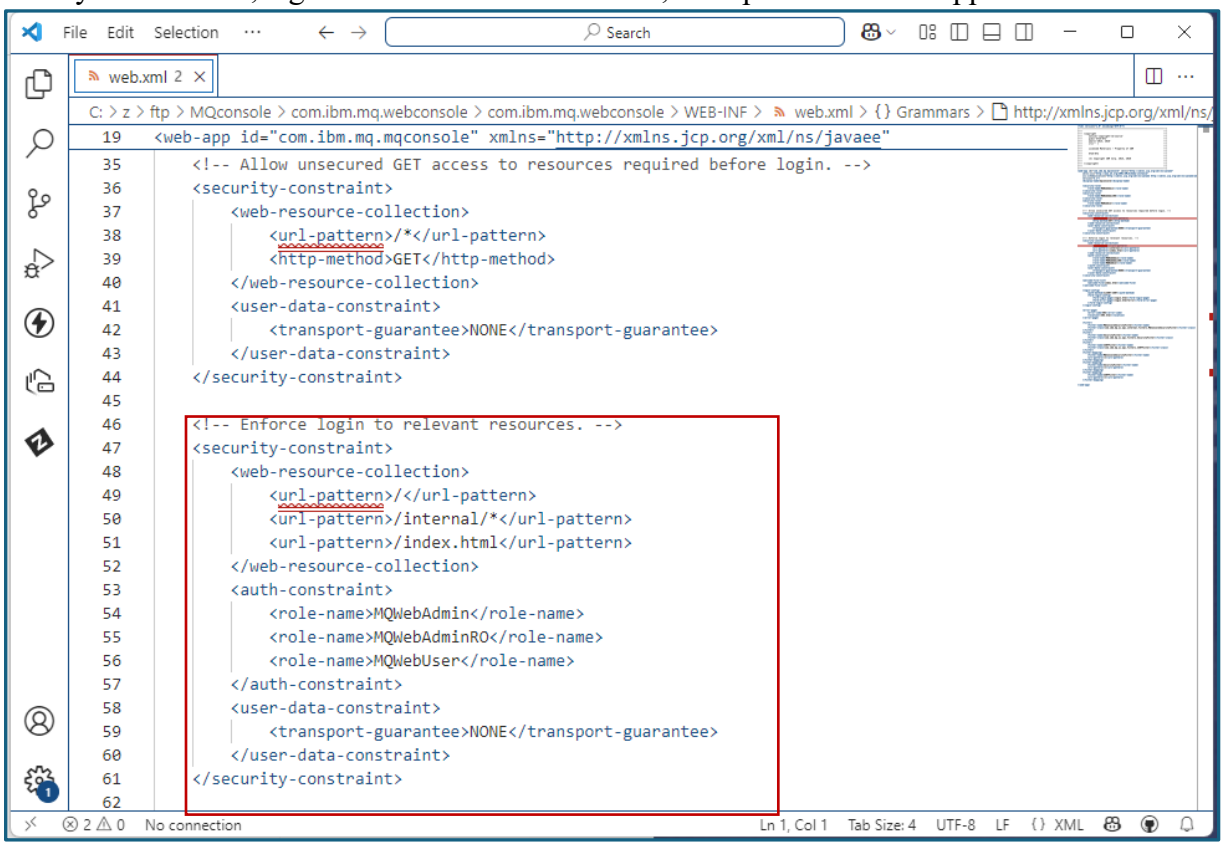
Applications deployed to Liberty servers packaged as **E**nterprise **Ar**chive (EAR) files or **W**eb **Ar**chive (WAR). These archive files is simply a compressed file containing the application artifacts, e.g., EJBs, JAR files, JSPs, libraries, etc.

Mitch Johnson (mitchj@us.ibm.com)

For the MQ Console, the MQ product provides the server XML that is needed to install the MQ Console application. The server XML configuration for the MQ Console is defined in *<enterpriseApplication/>* configuration elements which provide the location and name of the EAR files and the name the application.

```
 /usr/lpp/mqm/web/mq/etc/mqweb.xml
<enterpriseApplication id="com.ibm.mq.console"
    location="${wlp.install.dir}/mq/apps/com.ibm.mq.webconsole.ear"
    name="com.ibm.mq.console" . . .
</enterpriseApplication>
```

EAR and WAR files can be unzipped to expose the application artifacts. And in each EAR and WAR file there should be a *web.xml* file (see below) which contains the application defined security information, e.g. role names and constraints, etc. specific for that application.



From the above *web.xml* file, we can determine that the application uses three security roles, *MQWebAdmin*, *MQAdminRO* and *MQWebUser*. The MQ product also provides a server XML recommendation for the server's *<safCredentials/>* configuration element.

```
<safCredentials unauthenticatedUser="WSGUEST" profilePrefix="MQWEB"/>
```

Now we have the profilePrefix (*MQWEB*), we have the name of the application (*com.ibm.mq.console*) and the application role names (*MQWebAdmin, MQWEBAdminRO*, and *MQWebUser).*

Mitch Johnson (mitchj@us.ibm.com)

Using this information the EJBRoles that need to be defined to SAF are:

- MQWEB.com.ibm.mq.console.MQWebAdmin
- MQWEB.com.ibm.mq.console.MQWebAdminRO
- MQWEB.com.ibm.mq.console.MQWebUser

Define these EJBRoles to the SAF and grant READ access the identities and groups that need access.

Removing an identity's EJBRole Access

Once an identity has been authenticated SAF authorization information is cached locally in the server's memory. This information will remain cached in the memory until the period time specified by the *timeout* attribute of the <*authCache*> configuraton element expires, see section *The Liberty Authentication Cache <authCache>* on page 16.

Until the timeout occurs, if an identity's READ access to the **EJBROLE** profile is deleted or removed with a **PERMIT DELETE** command and a ***SETROPTS RACLIST(EJBROLE) REFRESH*** command, the identity does not immediately lose access to the **EJBROLE** resource on the next attempt. The same is true if group access checking is being used and the identity is removed from the group that has READ access to the **EJBROLE** resource. The reason is that the SAF access information for this identity remains cached in the server's memory. For an explanation as to why removing an identity from a group is not sufficient to remove the access, see the information at *Changing the group* membership of an identity on page 16.

A MVS MODIFY (F) command can be used to "clear" the server's cached information. Once this command is invoked on the server, the client's identity no longer has access to the **APPL** resource and no longer has access to the server. The MVS MODIFY command to do this for server *BAQSTRT* would be **F BAQSTRT,CACHE,CLEAR,AUTH**.

Liberty does provide a management bean (MBean), see section *18* on page 18 for information about this MBean and examples of how to execute it using cURL, Postman and the API Explorer.

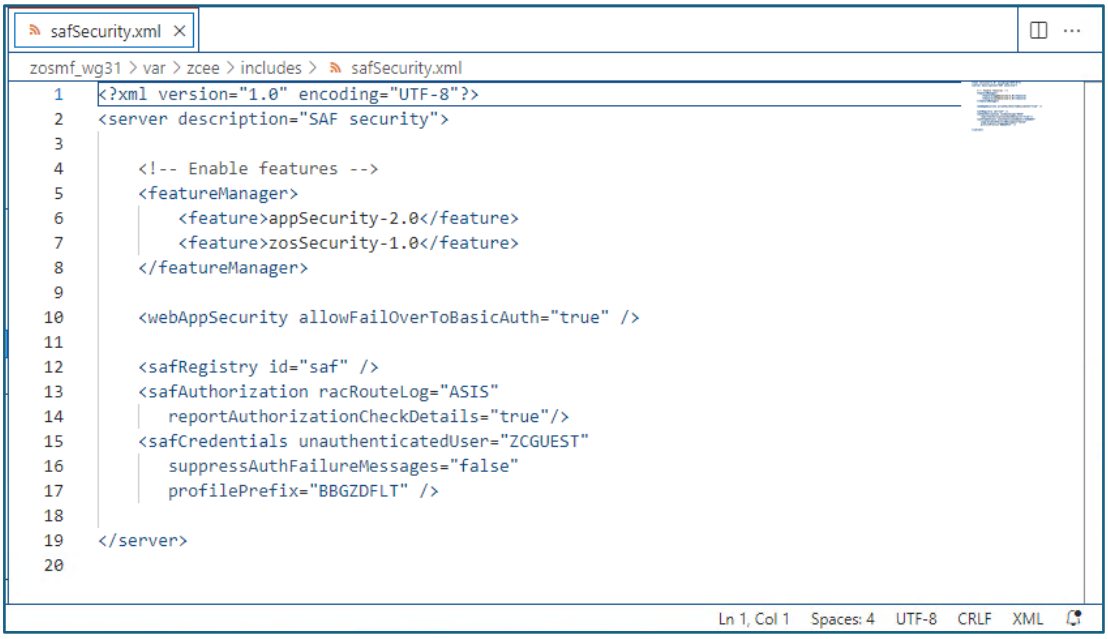
N.B *Invoking this command clears all the entries from the cache which means all subsequent will experience the overhead of making SAF service call for the next request.*

Mitch Johnson (mitchj@us.ibm.com)

SAF XML configuration elements

The Washington Systems Center recommends that the Liberty XML configuraton for SAF configuration elements be placed in a separate XML file and this file included in the *server.xml* file of each Liberty server when SAF security is required. Using a single file in a shared location means that only one file needs to be created and maintained.

Below is an example of a file for showing the configuration elements related to enabling SAF security. This file can be included in multiple server's configuration. These configuration elements are the minimum required for enabling SAF security in a Liberty server.

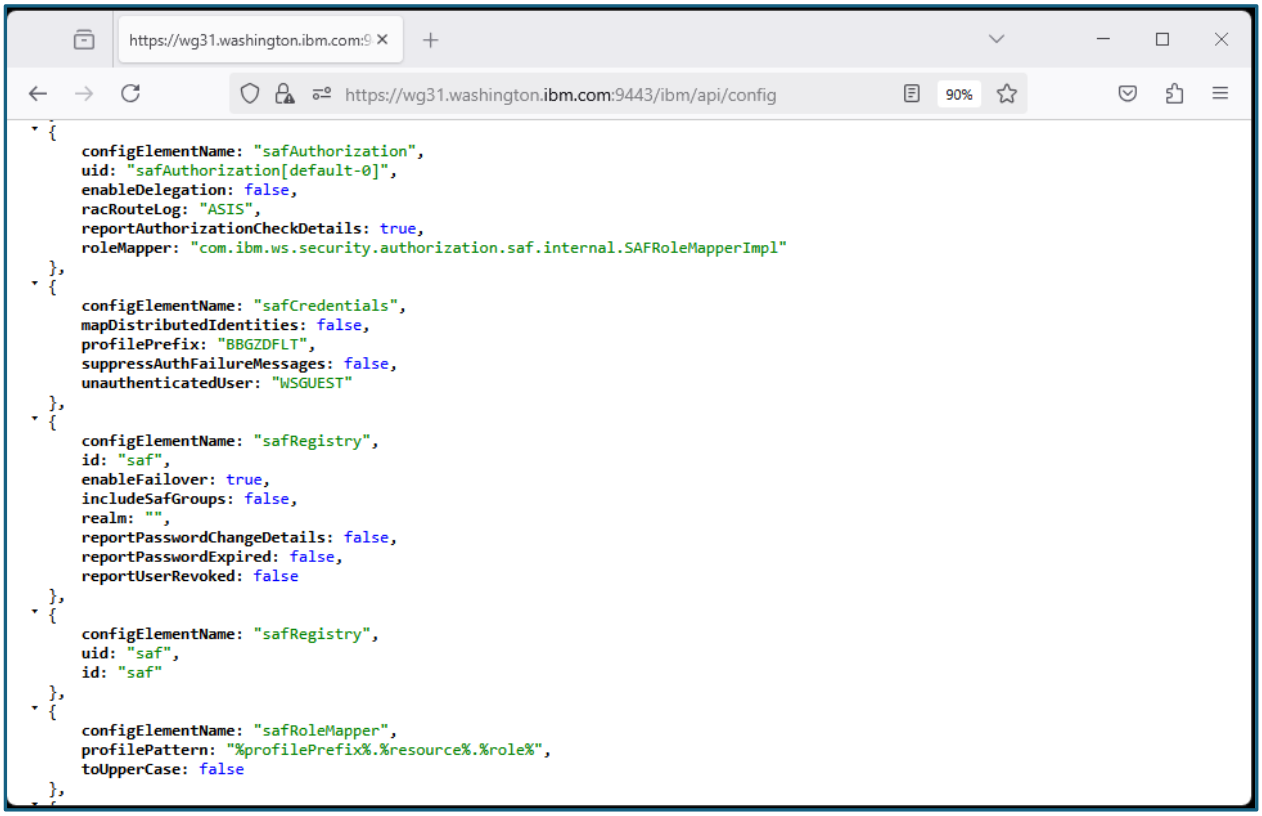


- Line 7 adds the Liberty feature for enabling z/OS security functions.
- Line 10 enables fail-over to basic authentication, otherwise basic authentication is not allowed
- Line 12 enables the use of the SAF registry for authenticating credentials.
- Line 13 enabling *reportAuthorizationCheckDetails* displays a message when the *unauthenticated user* attempts to access a protected resource.
- Line 13 setting *racRouteLog* to *ASIS* defers control of access logging to the manner specified in the resource profile.
- Line 14 identifies which SAF identity should be used for the *unauthenticated user.*
- Line 15 identifies the profile which should be used for isolation SAF EJBRole resources and the value that should be used for the SAF **APPL** resource. For maximum flexibility the value for the *profilePrefix* should be a variable that is set in the *bootstrap.property* file of a Liberty server.

N.B. This XML can be made more flexible by using variables values for the *unauthenticatedUser* and *profilePrefix* attributes. The values for these variables can be set in the *bootstrap.properties* file of each server.

Mitch Johnson (mitchj@us.ibm.com)

Note that once SAF security is enabled using the above XML configuration elements, the server will be configured with the SAF configuration shown below. Review the other configuration related to security and their default values.



For explanations of the attributes related to SAF security, see the configuraton information starting at URL https://www.ibm.com/docs/en/was-liberty/zos?topic=configuration-safauthorization

Mitch Johnson (mitchj@us.ibm.com)

Changing an identity's group access

Once an identity has been authenticated SAF authorization information is cached locally in the server's memory. This information will remain cached in the memory until the period time specified by the *timeout* attribute of the <*authCache*> configuraton element expires, see section *The* Liberty Authentication Cache <authCache> on page 16.

Until the timeout occurs, if an identity's membership in a group removed with a **CONNECT REMOVE** command, the identity does not immediately lose access to the **EJBROLE** resource on the next attempt. For an explanation as to why removing an identity from a group is not sufficient to remove the access, see the information at *Changing the group* membership of an identity on page 16.

A MVS MODIFY (F) command can be used to "clear" the server's cached information. Once this command is invoked on the server, the client's identity no longer has access to the **APPL** resource and no longer has access to the server. The MVS MODIFY command to do this for server *BAQSTRT* would be **F BAQSTRT,CACHE,CLEAR,AUTH**.

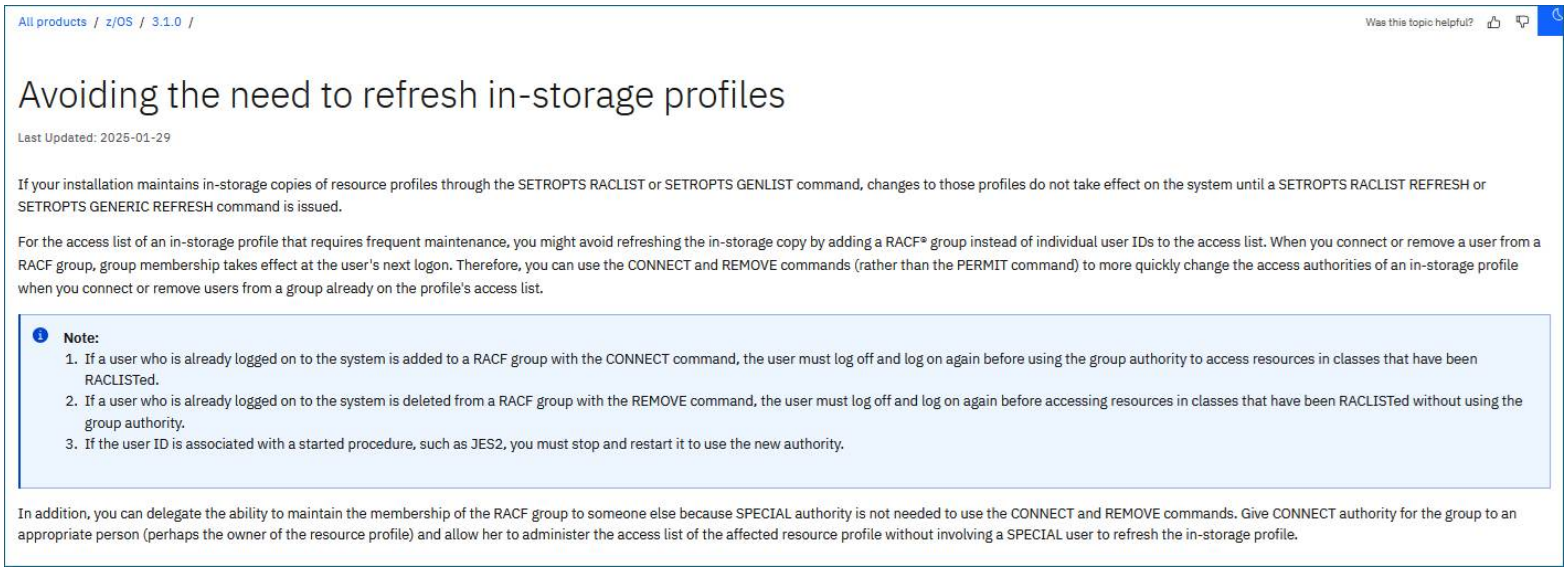
Liberty does provide a management bean (MBean), see section

Mitch Johnson (mitchj@us.ibm.com)

Using an MBean for clearing a server's instorage profile on page 18 for information about this MBean and examples of how to execute it using cURL, Postman and the API Explorer.

Changing the group membership of an identity

There is no SETROPTS command for refreshing the instorage profiles when group membership changes. This difference from other RACF resources is described in the **Note** at URL https://www.ibm.com/docs/en/zos/3.1.0?topic=groups-avoiding-need-refresh-in-storage-profiles, see below:



All products / z/OS / 3.1.0 /

Was this topic helpful?

Avoiding the need to refresh in-storage profiles

Last Updated: 2025-01-29

If your installation maintains in-storage copies of resource profiles through the SETROPTS RACLIST or SETROPTS GENLIST command, changes to those profiles do not take effect on the system until a SETROPTS RACLIST REFRESH or SETROPTS GENERIC REFRESH command is issued.

For the access list of an in-storage profile that requires frequent maintenance, you might avoid refreshing the in-storage copy by adding a RACF® group instead of individual user IDs to the access list. When you connect or remove a user from a RACF group, group membership takes effect at the user's next logon. Therefore, you can use the CONNECT and REMOVE commands (rather than the PERMIT command) to more quickly change the access authorities of an in-storage profile when you connect or remove users from a group already on the profile's access list.

Note:

1. If a user who is already logged on to the system is added to a RACF group with the CONNECT command, the user must log off and log on again before using the group authority to access resources in classes that have been RACLISTed.
2. If a user who is already logged on to the system is deleted from a RACF group with the REMOVE command, the user must log off and log on again before accessing resources in classes that have been RACLISTed without using the group authority.
3. If the user ID is associated with a started procedure, such as JES2, you must stop and restart it to use the new authority.

In addition, you can delegate the ability to maintain the membership of the RACF group to someone else because SPECIAL authority is not needed to use the CONNECT and REMOVE commands. Give CONNECT authority for the group to an appropriate person (perhaps the owner of the resource profile) and allow her to administer the access list of the affected resource profile without involving a SPECIAL user to refresh the in-storage profile.

Since stopping and restarting a server is not an option, an MVS MODIFY command is provided to invalidate the Liberty started task's local authentication cache. Subsequent authorization checks for an identity will obtain the current list of groups for that identity. The MVS MODIFY command for server *BAQSTRT* would be **F BAQSTRT,CACHE,CLEAR,AUTH.**

Some readers may question the difference in this behavior versus the behavior with CICS. This difference is because CICS has support for listening for RACF type 71 ENF events, see URL https://www.ibm.com/docs/en/zos/3.1.0?topic=signals-type-71-enf. When an identity is added or removed from a group member, RACF will send a type 71 ENF signal to registered listeners. See URL https://www.ibm.com/docs/en/cics-ts/6.x?topic=parameters-racfsync for details of the CICS system initialization parameter RACFSYNC. Liberty has no equivalent.

The Liberty Authentication Cache <authCache>

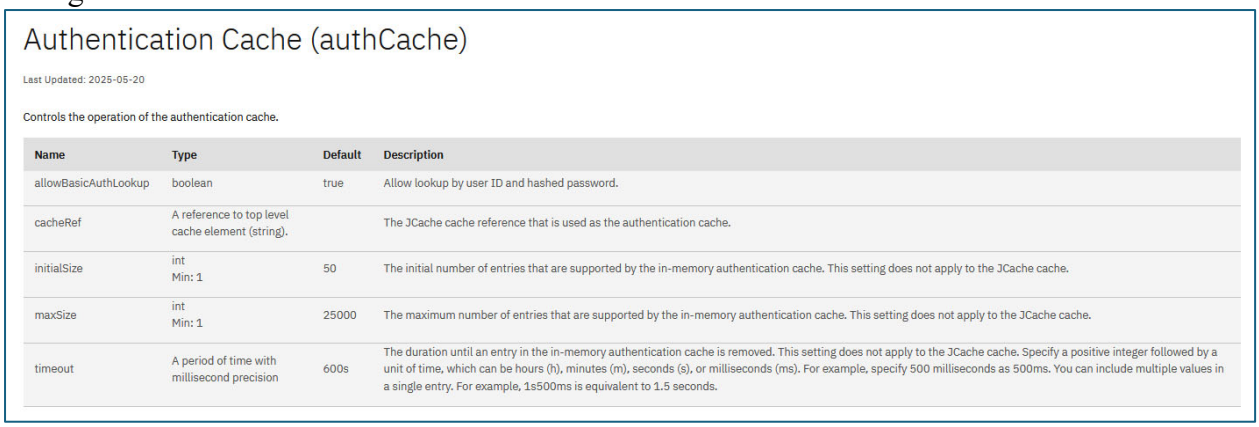
To improve performance once an identity is authenticated its SAF information obtain by making the SAF service calls is cached or stored in memory. Storing SAF information in memory eliminates the overhead of making additional SAF service calls for the next request by this user. SAF service call are relatively 'expensive' in instruction counts and are not zIIP eligible, so avoiding their overhead should be a goal.

The SAF information will remain in memory until a set period of inactivity has elapsed. A reap or scavenger process will periodically scan the cache and remove any stored SAF information on a per identity basis. The next time the identity is used for authentication purposes new SAF

Mitch Johnson (mitchj@us.ibm.com)

service calls are made and any update information is stored in the cache again. The default time for the inactivity interval is 10 minutes.

The Liberty <*authCache*> configuration element provides an administrator the means to configure the authentication cache.



Authentication Cache (authCache)

Last Updated: 2025-05-20

Controls the operation of the authentication cache.

Name	Type	Default	Description
allowBasicAuthLookup	boolean	true	Allow lookup by user ID and hashed password.
cacheRef	A reference to top level cache element (string).		The JCache cache reference that is used as the authentication cache.
initialSize	int Min: 1	50	The initial number of entries that are supported by the in-memory authentication cache. This setting does not apply to the JCache cache.
maxSize	int Min: 1	25000	The maximum number of entries that are supported by the in-memory authentication cache. This setting does not apply to the JCache cache.
timeout	A period of time with millisecond precision	600s	The duration until an entry in the in-memory authentication cache is removed. This setting does not apply to the JCache cache. Specify a positive integer followed by a unit of time, which can be hours (h), minutes (m), seconds (s), or milliseconds (ms). For example, specify 500 milliseconds as 500ms. You can include multiple values in a single entry. For example, 1s500ms is equivalent to 1.5 seconds.

The defaults should be acceptable but an administrator should be aware of their options to configure the authentication cache.

For more information about the authentication cache see URL https://www.ibm.com/docs/en/was-liberty/zos?topic=liberty-configuring-authentication-cache-in
And for more information about configuring the authentication cache in a Liberty server see URL https://www.ibm.com/docs/en/was-liberty/zos?topic=configuration-authcache

Mitch Johnson (mitchj@us.ibm.com)

Using an MBean for clearing a server's instorage profile

A management bean or MBean is function that can be invoked by a client to perform an administrative function. One of the many MBeans provided by Liberty is a function that will clear all the entries in the authentication cache. This is the MBean used by the MVS MODIFY command described earlier. Invoking MBeans does require the use of HTTPS and administrator authority for the EJBRole protecting this function.

WebSphere:service=com.ibm.websphere.security.authentication.cache.DeleteAuthCache

- **Management interface:** `com.ibm.websphere.security.authentication.cache.DeleteAuthCache`
- **Comments:** This MBean can be called to clear all users from the authentication cache. This MBean is available when the Application Security feature, version 2.0 or later, is configured.

This section describes the details of the required configuration changes and security resources needed to enable both JMX and REST access to a Liberty server endpoint.

Enable the required Liberty features

Liberty features *monitor-1.0* and *restConnector-2.0* are required to enable JMX clients like JConsole and REST clients like Postman and cURL to access a Liberty Server's MBean. Enable these features in the *<featureManager/>* section of the server' server XML configuration.

```
<!-- Enable features -->
<featureManager>
   <feature>appSecurity-2.0</feature>*
   <feature>zosSecurity-1.0</feature>*
   <feature>transportSecurity-1.0</feature>*
   <feature>apiDiscovery-1.0</feature>*
   <feature>monitor-1.0</feature>
   <feature>restConnector-2.0</feature>
</featureManager>
```

N.B. *Liberty servers on z/OS require the use of TLS for REST client access. It is my opinion that enabling TLS connection and other required security features is easiest done using SAF security. The features with the asterisk superscript are the that we automatically add when enabling SAF and TLS security.*

The *monitor-1.0* adds standard monitoring MBeans as well as MXBeans, the latter provide monitoring for specific runtime components. For more information about the *monitor-1.0* feature including the MXBeans added by this feature, see URL https://www.ibm.com/docs/en/was-liberty/zos?topic=environment-monitoring-monitor-10. The restConnector-2.0 feature adds remote REST client access to a set of administrative APIs over HTTP. For more information about the *restConnector-2.0* feature, see URL https://www.ibm.com/docs/en/was-liberty/zos?topic=features-admin-rest-connector-20.

Access to Liberty MBeans and REST administrative APIs is controlled by access to SAF EJBRole resources, see URL https://www.ibm.com/docs/en/was-liberty/zos?topic=liberty-mapping-management-roles-zos for more information.

Mitch Johnson (mitchj@us.ibm.com)

Below are examples of the commands that are used to define and permit access to the *Administrator* and *Reader* EJBRoles for Liberty administrative features.

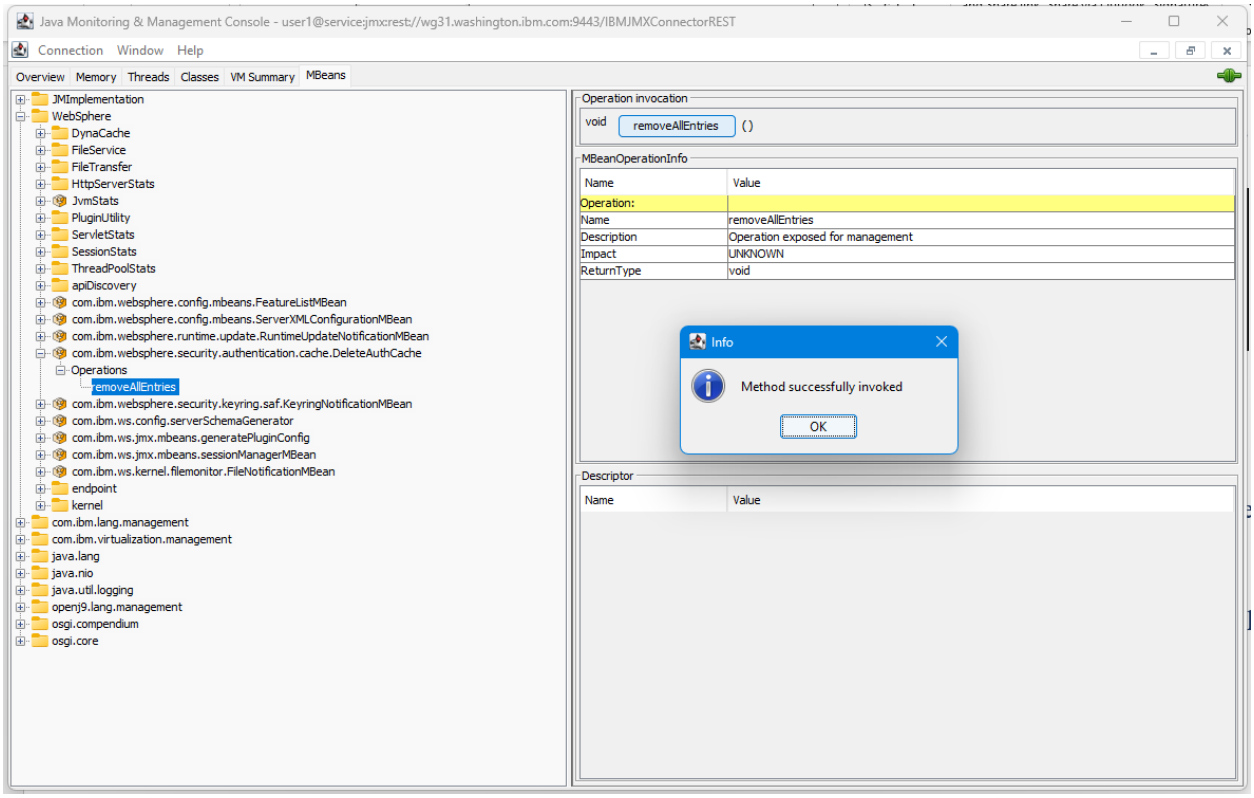
```
RDEFINE EJBROLE BBGZDFLT*.com.ibm.ws.management.security.resource.Administrator OWNER(SYS1)UACC(NONE)
RDEFINE EJBROLE BBGZDFLT*.com.ibm.ws.management.security.resource.Reader OWNER(SYS1) UACC(NONE)
PERMIT BBGZDFLT*.com.ibm.ws.management.security.resource.Administrator CLASS(EJBROLE) ID(ADMNUSRS) ACCESS(READ)
PERMIT BBGZDFLT*.com.ibm.ws.management.security.resource.Reader CLASS(EJBROLE) ID(OTHUSRS) ACCESS(READ)

SETR RACLIST(EJBROLE) REFRESH
```

JConsole

This MBean can also be invoked using the Java Console(JConsole) which included in the Java Developers Kit (JDK) or the Java Runtime Environment (JRE) on other platforms. See Monitoring a *z/OS Liberty server using JMX and REST clients* (URL https://ibm.biz/BdahXK) for instructions on how to enable JConsole support in a Liberty server.

Below is an example of involving this MBean to clear the SAF security cache.



Mitch Johnson (mitchj@us.ibm.com)

Client for URL (cURL)

Client for URL (cURL) is another popular command line interface for invoking this MBean. An advantage of using cURL is that this command can be placed in a command file, script or even invoked using JCL.

curl -X POST --header "Content-Type: application/json" -d"{}" --insecure --user USER1:user1 --location "https://wg31.washington.ibm.com:9455/IBMJMXConnectorREST/mbeans/**WebSphere:service=com.ibm.websphere.security.authentication.cache.DeleteAuthCache/operations/removeAllEntries**" -w " -HTTP CODE: %{http_code}"

Example of the output of a successful call.

C:\Users\948478897>curl -X POST --header "Content-Type: application/json" -d"{}" --insecure --user USER1:user1 --location "https://wg31.washington.ibm.com:9455/IBMJMXConnectorREST/mbeans/WebSphere:service=com.ibm.websphere.security.authentication.cache.DeleteAuthCache/operations/removeAllEntries" -w " -HTTP CODE: %{http_code}"

{"value":null,"type":null} -HTTP CODE: 200

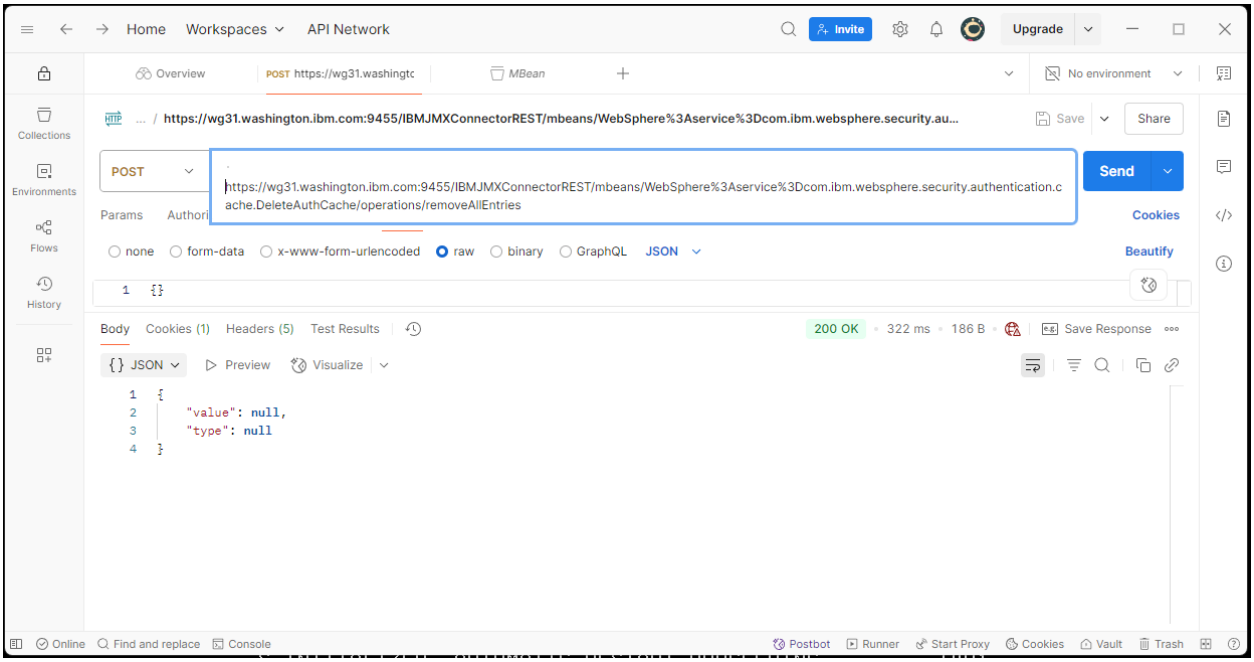
Below is an example of using Rocket Software's cURL for z/OS in JCL, see URL https://www.rocketsoftware.com/en-us/products/open-source for more information regarding using cURL on z/OS.

```
//EXPORT EXPORT SYMLIST=(*)
// SET CURL='/usr/lpp/rocket/'
//CURL PROC
//CURL EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
// PEND
//*****************************************************************
//*  STEP CURL - use cURL to clear the security cache
//*****************************************************************
//CURL EXEC CURL
//SYSTSIN  DD *,SYMBOLS=EXECSYS
BPXBATCH SH export CURL=&CURL; +
$CURL/curl/bin/curl -X POST --insecure --user USER1:user1 +
--header "Content-Type: application/json" -d"{}" --location +
"https://wg31.washington.ibm.com:9455/IBMJMXConnectorREST/+
mbeans/WebSphere:service=com.ibm.websphere.security.+
authentication.cache.DeleteAuthCache/operations/removeAllEntries" +
 -w " -HTTP CODE: %{http_code}"
```


Mitch Johnson (mitchj@us.ibm.com)

Postman

Below is an example of using Postman to invoke this MBean.



Mitch Johnson (mitchj@us.ibm.com)

Common SAF configuration Issues

This section provides information on identifying and resolving some common SAF configuration issues.

If a client is providing valid credentials and is receiving an *HTTP 401 Unauthorized* message, start problem determination by reviewing the *messages.log* file and look for messages like the ones shown in this section.

***N.B.** Since Liberty is running as an OMVS process, not all SAF errors will generate messages that will appear in the SYSLOG or the job logs of the server. When an authentication or authorization issue occurs, the server's messages.log file will usually be the best place to start to find the information needed to the resolve the issue.*

There two server XML configuration attributes which can be used to provide additional SAF related information. More diagnostic information will be written the SYSLOG and messages.log file when the suppressAuthFailureMessages attribute in the <safCredentials/> configuration element is set to false and the reportAuthorizationCheckDetails attribute in the <safAuthorization/> configuration element is set to true.

***N.B**. Setting **SAFLOG** to **Y** will provide additional useful SAF diagnostic information when an attempt is made to an unauthorized resource, for example, additional ICH408I messages in the SYSLOG and RACF services return and reason codes in the messages.log file, see URL https://www.ibm.com/docs/en/zos/3.1.0?topic=acee-return-reason-codes.*

No angel process available

A Liberty server must have access to an Angel to have access to z/OS privileged functions. Check to see if the server attempted to register with an Angel by looking at the server startup messages like the ones below.

```
CWWKB0101I: The angel process is not available.  No authorized services will be loaded.  The reason code is 4.
CWWKB0104I: Authorized service group KERNEL is not available.
CWWKB0104I: Authorized service group LOCALCOM is not available.
CWWKB0104I: Authorized service group PRODMGR is not available.
CWWKB0104I: Authorized service group SAFCRED is not available.
CWWKB0104I: Authorized service group TXRRS is not available.
CWWKB0104I: Authorized service group WOLA is not available.
CWWKB0104I: Authorized service group ZOSAIO is not available.
CWWKB0104I: Authorized service group ZOSDUMP is not available.
CWWKB0104I: Authorized service group ZOSWLM is not available.
CWWKB0104I: Authorized service group CLIENT.WOLA is not available.
```

These messages mean that the server's Angel process is not active.

The resolution is to start the Angel process.

Mitch Johnson (mitchj@us.ibm.com)

N.B. *By default, if an Angel is not available the server will continue the startup but without access to SAF services. Lack of access to an Angel may not manifest until hours later. To stop the server from even starting in situation, use the Java directive* ***com.ibm.ws.zos.core.angelRequired****. When this directive is set to true the server will terminate immediately if the required Angel is not available.*

Not authorized to register with the angel process

A Liberty server must have the authority to register with an Angel. Check to see if the server is authorized to register with the Angel. Review the server startup messages for messages like the ones below.

```
CWWKB0118W: This server is not authorized to connect to the angel process. No authorized services will be loaded.
CWWKB0104I: Authorized service group KERNEL is not available.
CWWKB0104I: Authorized service group LOCALCOM is not available.
CWWKB0104I: Authorized service group PRODMGR is not available.
CWWKB0104I: Authorized service group SAFCRED is not available.
CWWKB0104I: Authorized service group TXRRS is not available.
CWWKB0104I: Authorized service group WOLA is not available.
CWWKB0104I: Authorized service group ZOSAIO is not available.
CWWKB0104I: Authorized service group ZOSDUMP is not available.
CWWKB0104I: Authorized service group ZOSWLM is not available.
CWWKB0104I: Authorized service group CLIENT.WOLA is not available.
```

This set of messages occur at server startup and occur because the server's SAF identity does not have READ access to the **SERVER** resource (e.g. BBG.ANGEL) protecting the default or no name Angel. Note that if an Angel is not available the server will continue the startup but without access to SAF services. Issues caused by the lack of access may not manifest until a client tries to connect to the server and requires authentication.

The resolution is to permit READ access to the SAF **SERVER** resource protecting the Angel to the server's SAF identity.

N.B. *Sometimes the Angel may be available but access to a required privileged service is not. Setting Java directive* ***com.ibm.ws.zos.core.angelRequiredServices*** *can also be used to prevent the server from starting in this situation. To stop the server from staring in this situation. add Java directive com.ibm.ws.zos.core.angelRequiredServices=SAFCRED to a bootstrap.properties file. This will cause the server to shut down if access to SAFCRED is not available. For more information on this directive, see URL https://www.ibm.com/docs/en/was-liberty/zos?topic=zos-process-types.*

Mitch Johnson (mitchj@us.ibm.com)

The server's SAF identity does not have access to the security domain

• The next scenario occurs after server startup and only when the first authentication attempt is made. This occur because the *server's SAF identity* does not have READ access to the **SERVER** resource (e.g., BBG.SECPFX.*profilePrefix*) limiting the scope of the server's security domain identified by the server's SAF APPL, e.g., the *profilePrefix* attribute.

The Angel related startup messages look normal
CWWKB0122I: This server is connected to the default angel process.
CWWKB0104I: Authorized service group KERNEL is available.
CWWKB0104I: Authorized service group LOCALCOM is available.
CWWKB0104I: Authorized service group PRODMGR is available.
CWWKB0104I: Authorized service group SAFCRED is available.
CWWKB0104I: Authorized service group TXRRS is available.
CWWKB0104I: Authorized service group WOLA is available.
CWWKB0104I: Authorized service group ZOSAIO is available.
CWWKB0104I: Authorized service group ZOSDUMP is available.
CWWKB0104I: Authorized service group ZOSWLM is available.
CWWKB0104I: Authorized service group CLIENT.WOLA is available.
Normal server startup messages appear until a user tries to authenticate
CWWKS2960W: Cannot create the default credential for SAF authorization of unauthenticated users. All authorization checks for unauthenticated users will fail. The default credential could not be created due to the following error: CWWKS2909E: A SAF authentication or authorization attempt was rejected because the server is not authorized to access the following SAF resource: APPL-ID BBGZDFLT. Internal error code 0x03008108.
CWWKS2932I: The authorized version of the SAF user registry is activated. Authentication will proceed using authorized native services.
CWWKS2930W: A SAF authentication attempt using authorized SAF services was rejected because the server is not authorized to access the APPL-ID BBGZDFLT. Authentication will proceed using unauthorized SAF services.
Since SAF authentication has failed, Liberty now attempts to use OMVS to authenticate the user's credentials, but OMVS authentication has not been configured
CWWKS2933E: The username and password could not be checked because the BPX.DAEMON profile is active, and the address space is not under program control.
FFDC1015I: An FFDC Incident has been created: "com.ibm.ws.security.registry.RegistryException: Unix System Service __passwd failed for user USER1 with errno 157 (EMVSERR) and errno2 x90c02af
com.ibm.ws.security.registry.saf.internal.SAFRegistry 135" at ffdc_25.06.05_16.08.47.0.log
CWWKS1100A: Authentication did not succeed for user ID USER1. An invalid user ID or password was specified.
CWWKS2933E: The username and password could not be checked because the BPX.DAEMON profile is active, and the address space is not under program control.
CWWKS1100A: Authentication did not succeed for user ID USER1. An invalid user ID or password was specified

In this case the messages in the messages.log file as well as message that appears in the SYSLOG are misleading (see below).

BPXP015I HFS PROGRAM /usr/lib/java_runtime/libifaedjreg64.so IS NOT MARKED PROGRAM CONTROLLED.
BPXP014I ENVIRONMENT MUST BE CONTROLLED FOR DAEMON (BPX.DAEMON) PROCESSING.

The resolution is to grant access to the security domain to the server's SAF identity.

Mitch Johnson (mitchj@us.ibm.com)

N.B. *The root cause of this issue is not because the unauthenticated identity does not have access to the APPL resource. Nor is it related to a HFS program not being marked program controlled. The root cause is lack of access to* ***SERVER*** *resource BBG.SECPFX.BBGZDFLT.*

Note that when <safCredentials/> attribute suppressAuthFailureMessages is set to false, RACF messages ICH408I messages are written to the SYSLOG clearing indicating that this lack of access to the ***SERVER*** *profile is the root cause of the issue.*

```
ICH408I USER(LIBSERV ) GROUP(LIBGRP  ) NAME(LIBERTY SERVER
  BBG.SECPFX.BBGZDFLT CL(SERVER  )
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
ICH408I USER(LIBSERV ) GROUP(LIBGRP  ) NAME(LIBERTY SERVER
  BBG.SECPFX.BBGZDFLT CL(SERVER  )
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
ICH408I USER(LIBSERV ) GROUP(LIBGRP  ) NAME(LIBERTY SERVER
  BBG.SECPFX.BBGZDFLT CL(SERVER  )
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
BPXP015I HFS PROGRAM /usr/lib/java_runtime/libifaedjreg64.so IS NOT MARKED PROGRAM CONTROLLED.
BPXP014I ENVIRONMENT MUST BE CONTROLLED FOR DAEMON (BPX.DAEMON) PROCESSING.
BPXP018I THREAD 13D6100000000000, IN PROCESS 67113642, ENDED
WITHOUT BEING UNDUBBED WITH COMPLETION CODE 40222000
, AND REASON CODE 00000000.
```

The client identity is not in the server security domain

• The next scenario occurs after server startup and only when the identity USER3 makes an authentication attempt is made. This occur because the *SAF identity* USER3 does not have READ access to the **APPL** resource (e.g., BBGZDFLT) and therefore not in the scope of the server's security domain identified by the server's SAF APPL, e.g., the *profilePrefix* attribute.

The Angel related startup messages look normal
CWWKB0122I: This server is connected to the default angel process.
CWWKB0104I: Authorized service group KERNEL is available.
CWWKB0104I: Authorized service group LOCALCOM is available.
CWWKB0104I: Authorized service group PRODMGR is available.
CWWKB0104I: Authorized service group SAFCRED is available.
CWWKB0104I: Authorized service group TXRRS is available.
CWWKB0104I: Authorized service group WOLA is available.
CWWKB0104I: Authorized service group ZOSAIO is available.
CWWKB0104I: Authorized service group ZOSDUMP is available.
CWWKB0104I: Authorized service group ZOSWLM is available.
CWWKB0104I: Authorized service group CLIENT.WOLA is available.
Normal server startup messages appear until the user tries to authenticate
E CWWKS2907E:SAF Service IRRSIA00_CREATE did not succeed because user user3 has insufficient authority to access APPL-ID BBGZDFLT. SAF return code 0x00000008. RACF return code 0x00000008. RACF reason code 0x00000020.
A CWWKS1100A: Authentication did not succeed for user ID user3. An invalid user ID or password was specified.

Mitch Johnson (mitchj@us.ibm.com)

The resolution is to grant access to the security domain to the client's identity, e.g., permit the identity to the security domain's **APPL** resource.

Note the messages below appeared in the SYSLOG messages. The significance of these messages is that when the identity was found not to be in the server's security domain, the security context switch did happen. The thread continued to run under the authority of the unauthenticated user(WSGUEST). The request continued until the SAF check for the **EJBRole** protecting this application failed. This is one of the reason the unauthenticated identity should be protected and no access to SAF resources other than the APPL resource protecting the server.

```
ICH408I USER(USER3   ) GROUP(SYS1    ) NAME(####################)
  BBGZDFLT CL(APPL    )
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
ICH408I USER(USER3   ) GROUP(SYS1    ) NAME(####################)
  BBGZDFLT CL(APPL    )
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
ICH408I USER(WSGUEST ) GROUP(WSGUESTG) NAME(UNAUTHENTICATED USER)
  BBGZDFLT.db2API.Staff CL(EJBROLE )
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

Tracing SAF service in Liberty

Occasionally tracing SAF authentication/authorization issues may be required. Below is an example of the values of the *traceSpecification* attribute I used in an <*logging/*> configuration XML element.

Adding an <*include/*> statement for a file(e.g. *safTrace.xml*) containing this logging statement and doing a modify config refresh MVS command (e.g. **F BAQSTRT,REFRESH,CONFIG**) will activate a trace of these components. By default, the trace file will be written to the server's logs directory. Stopping the trace can be done by removing the <*include/*> statement and repeating the modify command.

safTrace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="security trace">

<logging traceSpecification=
        "com.ibm.ws.security.*=all:
         SSLChannel=all:
         SSL=all"/>
</server>
```


Mitch Johnson (mitchj@us.ibm.com)

SERVER resources related to accessing z/OS privileges

Below is an example of the commands that can be used to define the **SERVER** resource profiles related to accessing z/OS privileges.

```
/* SERVER profile for the default (no name) Angel
RDEFINE SERVER BBG.ANGEL UACC(NONE) OWNER(SYS1)
PERMIT  BBG.ANGEL CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
/* SERVER profile for the named Angel (BBGZDFLT)
RDEFINE SERVER BBG.ANGEL.BBGZDFLT UACC(NONE) OWNER(SYS1)
PERMIT  BBG.ANGEL.BBGZDFLT CLASS(SERVER) ACCESS(READ) ID(LIBSERV)

/* SERVER profile for the security profilePrefix (BBGZDFLT)
RDEFINE SERVER BBG.SECPFX.BBGZDFLT UACC(NONE)
PERMIT  BBG.SECPFX.BBGZDFLT  CLASS(SERVER) ACCESS(READ)ID(LIBSERV)

/* SERVER profile for the authorized module BBGZSAFM
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
/* SAF authorized user registry services and SAF authorization services (SAFCRED)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.SAFCRED UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.SAFCRED  CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
/* WLM services (ZOSWLM)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSWLM UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.ZOSWLM  CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
/* RRS transaction services (TXRRS)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.TXRRS UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.TXRRS CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
/* SVCDUMP services (ZOSDUMP
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSDUMP UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.ZOSDUMP CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
/* Server optimized local adapter services
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.WOLA UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM.WOLA CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.LOCALCOM UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM.LOCALCOM CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
/* IFAUSAGE services (PRODMGR)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.PRODMGR UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM.PRODMGR CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
/* AsyncIO services (ZOSAIO)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSAIO UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM.ZOSAIO CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
/* SERVER profile for the authorized client module BBGZSCFM
RDEFINE SERVER BBG.AUTHMOD.BBGZSCFM UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSCFM  CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
/* Client optimized local adapter services:
RDEFINE SERVER BBG.AUTHMOD.BBGZSCFM.WOLA UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSCFM.WOLA  CLASS(SERVER) ACCESS(READ) ID(LIBSERV)

/* Don't forget to refresh the instorage profiles
SETROPTS RACLIST(SERVER) REFRESH
```

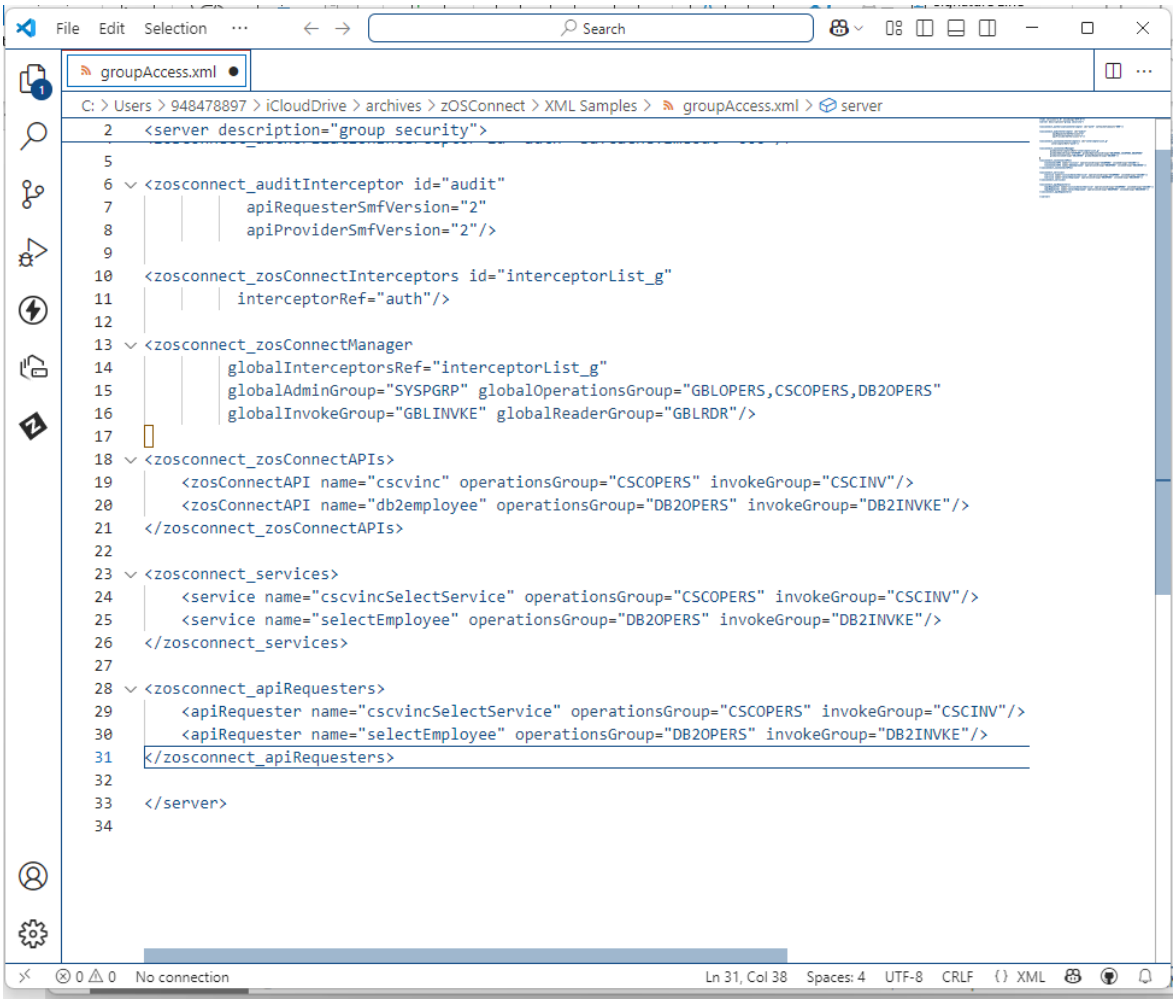

Mitch Johnson (mitchj@us.ibm.com)

z/OS Connect OpenAPI2 group SAF checks

A z/OS Connect OpenAPI2 server has an option where a third SAF resource (group membership) controls access to APIs, services and/or API requester artifacts. The group membership check is optional and requires the z/OS Connect OpenAPI2 authorization interceptor to be enabled.

If the mapped identity of the is not a member of one of the groups in the group list for a specific resource or function, the resource request is rejected with an HTTP 403 code.

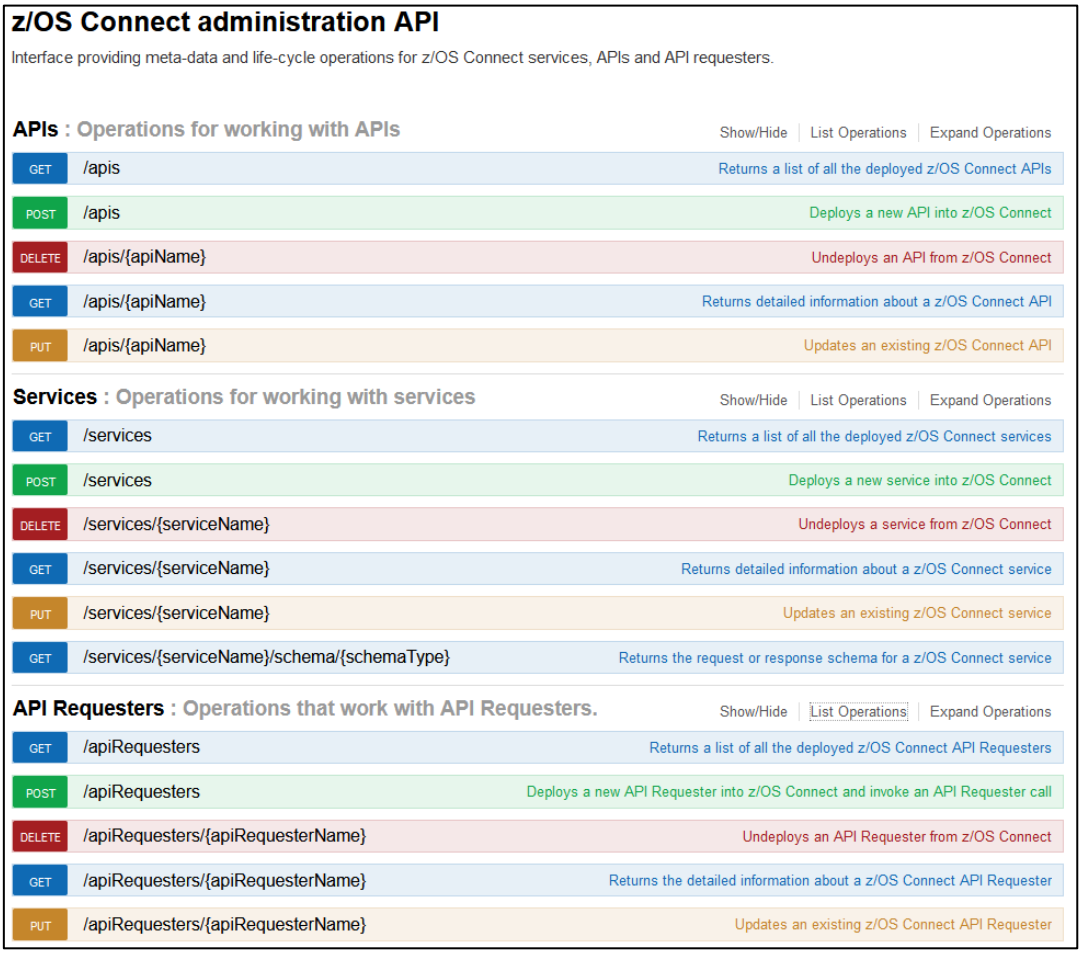
The best way to describe how this security works is to use an example. Using the server XML configuration below and the SAF groups references in the XML.



- Members of groups SYSPGRP, GBLOPERS,CSCOPERS, DB2OPERS and GBLRDR can connect to a z/OS server from the API toolkit.
- Members of groups SYSPGRP, GBLOPERS,CSCOPERS, and DB2OPERS can deploy new artifacts.
- Only members of groups CSCOPERS and DB2OPERS can manage their respective artifacts after they are deployed.
- Only members of groups CSCINV and Db2INVKE can execute their respective APIs, services or API requesters artifacts.

Mitch Johnson (mitchj@us.ibm.com)

This behavior can be explained using the figure below which shows the z/OS Connect OpenAPI2 RESTful administrative APIs.



- To be able to connect from the API toolkit, the client identity must be able to list all the APIs and services (GET method). Only members in the global administrators, global operations and global reader groups would be able to connect and list all the resources deployed to the server.
- Resources are deployed using the POST method and note there is no name of a resource, e.g. *apiName, serviceName* or *apiRequesterName*, in the URI path. To deploy a new resource requires an identity to be in a global administration or global operations group since there is nothing to use to trigger a match of a protected resource during deployment.
- The other administrative APIs (delete, change the state or get information about a specific resource) include a z/OS Connect resource name in the URI path. So, if there is an explicit XML element protecting that resource, then only members of the groups listed in that XML element for a function are authorized to perform that function for that resource. If an XML element exists for a resource, but the function being requested is not explicitly protected in the element, protection falls back to the global groups. This is the request for invoking an API, and there not an *invokeGroup* attribute in the resource's XML element, the *globalInvoke* attribute group provides protection.

Mitch Johnson (mitchj@us.ibm.com)