EntireX Security under z/OS

This chapter introduces EntireX Security under z/OS through overviews of the functionality and components of EntireX Security. The location where Broker Kernel is installed determines the functionality made available for EntireX Security. This chapter covers the following topics:

- Introduction
- EntireX Security for EntireX Broker
- Configuration Options for Broker
- Resource Profiles in EntireX Security

Note:
Installation of the security software is described under *Installing EntireX Security under z/OS*.

Introduction

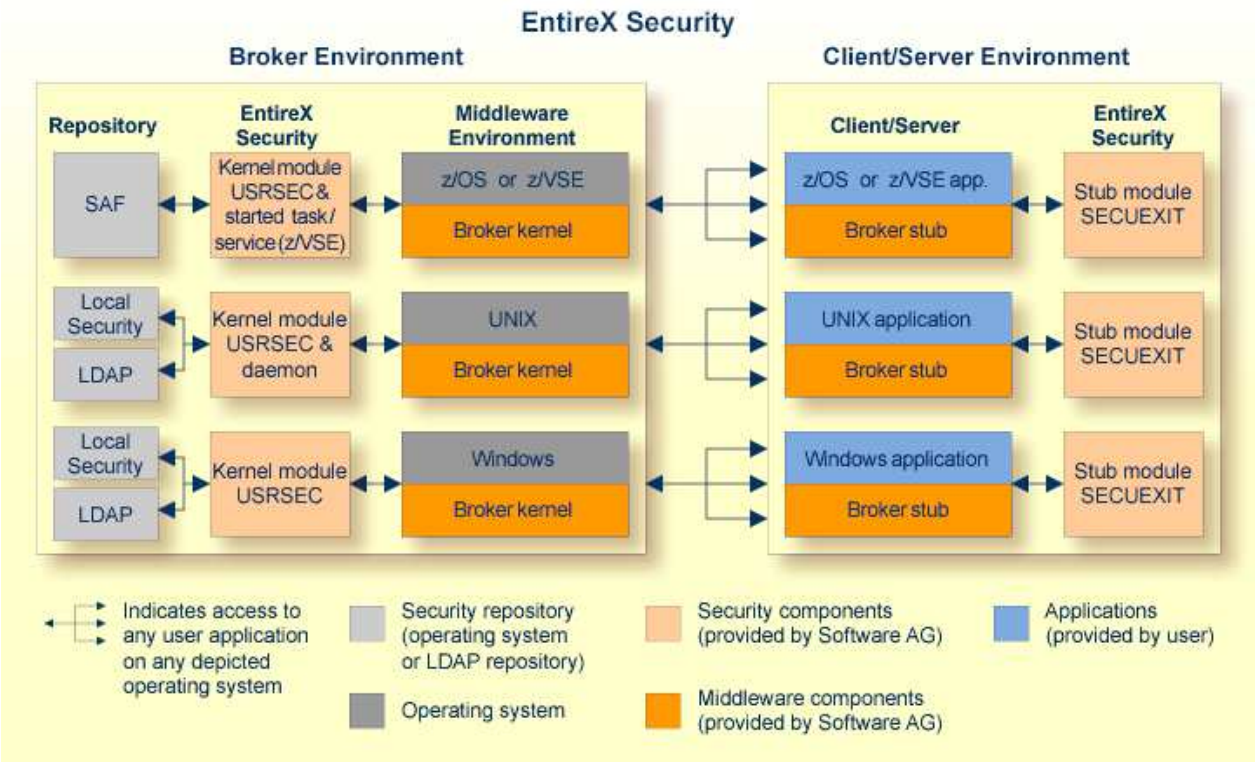
Functionality of EntireX Security

This table lists the security functionality available with EntireX Security running Broker Kernel under the respective operating system. See also *Configuration Options for Broker*.

Security Functionality	z/OS	UNIX	Windows	BS2000/OSD	z/VSE	Comment
Authentication of user	Yes	Yes	Yes	Yes	Yes	Verify User ID password.
User password change	Yes	No	No	No	No	
LDAP authentication	No	Yes	Yes	No	No	Authenticate using LDAP repository.
Trusted user ID	Yes	No	No	No	No	Trusted computer base, avoiding plain text password.
Verified client user ID	Yes	No	No	Yes	Yes	Provide verified identity of client to server.
Authorization of client request	Yes	No	No	No	No	
Authorization of server register	Yes	No	No	No	No	
Authorize IP connection	Yes	No	No	No	No	
Authorization rules	No	Yes	Yes	No	No	Check rules stored in an LDAP repository. These rules are maintained using an agent of System Management Hub, and are independent of the LDAP authentication mechanism. **Note:** These rules can be stored either in the same or a different LDAP repository.
Encryption of application data	Yes	Yes	Yes	No	Yes	RC4-compatible algorithm.
Guaranteed encryption	Yes	Yes	Yes	No	Yes	Allows administrator to require encryption for specific services.
SSL	Yes	Yes	Yes	No	No	Industry standard encryption mechanism.

EntireX Security Components

This diagram depicts the location where the Broker kernel must be installed and where the Broker stubs can be installed. It also depicts the location of the security components of the kernel and stubs of Broker.


EntireX Security
Broker Environment
Client/Server Environment
Repository
EntireX Security
Middleware Environment
Client/Server
EntireX Security
SAF
Kernel module USRSEC & started task/ service (z/VSE)
z/OS or z/VSE
Broker kernel
z/OS or z/VSE app.
Broker stub
Stub module SECUEXIT
Local Security
LDAP
Kernel module USRSEC & daemon
UNIX
Broker kernel
UNIX application
Broker stub
Stub module SECUEXIT
Local Security
LDAP
Kernel module USRSEC
Windows
Broker kernel
Windows application
Broker stub
Stub module SECUEXIT
Indicates access to any user application on any depicted operating system
Security repository (operating system or LDAP repository)
Operating system
Security components (provided by Software AG)
Middleware components (provided by Software AG)
Applications (provided by user)

EntireX Security for EntireX Broker

This section presents EntireX Security for EntireX Broker under the z/OS operating system covers the following topics:

- Introduction
- Considerations for Mainframe Application Components

See also *Security-specific Attributes* under *Broker Attributes* and *Operator Commands*.

Introduction

EntireX Broker acts as an agent to make the creation and operation of client/server applications simpler and more effective. Any number of server applications can be built for use by any number of clients. EntireX Security allows you to protect your server applications and clients independently.

Clients and servers are authenticated by user ID and password on their first contact with the system. Authorization is sought for specific server applications before a client is allowed access. This enables control of your distributed application systems - at both the application level and the client level.

Authorization is also required for server applications to register services. Unauthorized servers can be intercepted when trying to register. Facilities exist to establish connection authorization when client and server applications first establish contact with the Broker.

Considerations for Mainframe Application Components

Application components running in a mainframe environment that communicate using EntireX Broker interact with EntireX Security in the following ways:

- No password is required for applications executing under mainframe where the trusted user ID option is implemented. This is true for both client and server application components. EntireX Security automatically acquires the logged-on user ID. Utilizing the "trusted" user ID avoids having to supply the password again. It also requires customers to configure security for their mainframe environment(s), for example ensuring that the CICS system is protected by RACF.
- Applications can override the trusted user ID by supplying a valid user ID/password combination in the ACI control block. This causes EntireX Security to ignore the trusted user ID in favor of the supplied credentials. Applications must therefore ensure that they do not assign an incorrect user ID or spurious password to the ACI control block, where "trusted" user ID is implemented. The `CLIENT-ID` as conveyed in the ACI to the server component of the application now represents the client’s verified user ID, derived either from valid user ID/password credentials or from trusted user ID itself.

Configuration Options for Broker

This section describes the parameters for configuring EntireX Security under z/OS. You may either accept or modify the default settings which are specified in the Broker attribute file `DEFAULTS=SECURITY`. Always check installation options against the corresponding resource profile. See *Resource Profiles in EntireX Security*.

This section covers the following topics:

- Authentication
- Trusted User ID
- Request Authorization
- Request Authorization for Command and Info Services
- Ignore Security Token
- Guaranteed Encryption / Decryption Mechanism
- Authorize IP Connection
- Access to Undefined Resources
- Alternate Resource Class/Type Names
- Length of Resource Class/Type Profile
- Password to Uppercase
- Security Level
- Verified Client User ID
- Security Node
- Client RPC Authorization
- Considerations for Mainframe Natural Application Components

Authentication

Authentication is mandatory and performed for both client and server applications based on user ID and password. First contact with the Broker results in the host security system being referenced. If authentication fails, access is denied and the application is informed with a suitable error message.

It is the responsibility of both client and server applications to supply a valid user ID and password when calling the Broker. The user ID must be supplied with all commands. The password is required only for the first command and should not be supplied subsequently, except when executing multiple instances of the same application.

Authentication expires after a period of non-activity after which it must be repeated. User ID and password must be resupplied before further access is possible. The time limits `CLIENT-NONACT` and `SERVER-NONACT` determine these timeout periods and are defined in the Broker attribute file.

Note:
Applications must not assign a password to the ACI control block if they intend to use trusted user ID. This applies to all applications, including EntireX RPC Server.

Trusted User ID

This allows z/OS-based applications to communicate securely without having to supply user ID and password. Activate this option by specifying the following parameter in job member `WAL`*`vrs`*`.JOBS(SAFI010)`:

`TRUSTED-USERID=N`	Require user ID/password for z/OS application components.
`TRUSTED-USERID=Y`	Leverage trusted user ID mechanism for z/OS applications.

Make sure the Adabas Security Interface is enabled by specifying the following parameter in the source assembled by job member `WAL`*`vrs`*`.JOBS(ASMGBLS)`:

`SAF=YES`	Adabas Security Interface in use.

Request Authorization

Clients request distributed processing using the `SEND` command, indicating the class, name and service to be invoked. The Broker transmits the request to the server only if the client has access to the relevant resource profile. Similarly, servers are allowed to `REGISTER` services only if the server has access to the resource profile. The default profile make-up comprises class.name.service of the service. The following system parameters will modify the resource profile if required:

`INCLUDE-CLASS`	`={YES,NO}`	Include Class in resource check.
`INCLUDE-NAME`	`={YES,NO}`	Include Name in resource check.
`INCLUDE-SERVICE`	`={YES,NO}`	Include Service in resource check.

Note:
At least one option must be "YES" for authorization to be performed.

For example: if `INCLUDE-CLASS=YES`, `INCLUDE-NAME=YES` and `INCLUDE-SERVICE=YES`, the structure of the resource profile checked is:

<class_name>.<server_name>.<service_name>

But with `INCLUDE-CLASS=YES`, `INCLUDE-NAME=NO` and `INCLUDE-SERVICE=YES`, the profile would look like:

<class_name>.<service_name>

Alternatively, with INCLUDE-CLASS=NO, INCLUDE-NAME=YES and INCLUDE-SERVICE=YES, the resource profile to be checked is:

<server_name>.<service_name>

Clients require READ access to obtain processing from a server application and servers require CONTROL access in order to REGISTER successfully, otherwise the command is rejected.

Authorization checks are also performed for publish-and-subscribe processing. Subscribers are allowed to SUBSCRIBE only if they have READ access to the resource profile representing the TOPIC to which they are subscribing. A publisher requires CONTROL access in order to successfully PULBLISH to a topic.

Discrete or generic resource profiles can be defined for this purpose.

Note:
If you set SECURITY-NODE, the Broker ID is used as a prefix for all authorization checks.

Request Authorization for Command and Info Services

If you are using one of the following RPC servers with a broker protected by, for example, RACF or CA Top Secret, at least READ access is required to resources SAG.ETBCIS.INFO and SAG.ETBCIS.CMD.

- CICS ECI RPC Server
- IMS Connect RPC Server
- Micro Focus RPC Server
- RPC-ACI Bridge
- WebSphere MQ RPC Server
- Java RPC Server
- XML/SOAP RPC Server

Ignore Security Token

A security token is generated by EntireX. It is the responsibility of the application to clear the security token before making the first call and thereafter to maintain the contents of the field for the duration of communication for the user.

If validation of security token is not required - for example, where applications or packages do not maintain the security token in the ACI control block - this option may be switched off. The default setting is "NO" (do not ignore Security Token).

IGNORE-STOKEN={YES,NO}	Do not ignore Security Token.

Guaranteed Encryption / Decryption Mechanism

EntireX Security ensures message encryption consistency regardless of any configuration errors. This means, if the relevant assembly parameters or environment variables are incorrectly or inconsistently specified, the integrity of the Broker message is honored. This feature requires upgrade of all EntireX Security Broker stub and kernel components in all places.

See Broker attribute `ENCRYPTION-LEVEL` and control block field `ENCRYPTION-LEVEL`.

Authorize IP Connection

Communication between distributed application components and the Broker via TCP/IP can be subject to an authorization check at connection time. Define the following system parameter if this option is required:

`CHECK-IP-ADDRESS={YES,NO}` Authorize IP connection.

Access to Undefined Resources

The normal mode of operation is to prevent access to resources not defined to the security system. Profiles representing services are added to the security repository with either a default access or by granting access to specific users and groups. Access to undefined resources can be permitted using the following system parameter:

`UNIVERSAL={YES,NO}` Allow access to undefined resources.

Note:
This option does not permit access to resources defined with universal access "none". See also note on defining resources to ACF.

Alternate Resource Class/Type Names

By default, the resource class/type `NBKSAG` is used when performing authorization checks. The name of an alternate resource class can be specified using the following system parameter:

`SAF-CLASS=NBKSAG` Resource class for Broker.

Length of Resource Class/Type Profile

By default, the maximum length of the resource class/type profiles is 80 characters when performing authorization checks. Longer resource profiles can be checked by increasing the maximum resource profile length as follows. Make sure you also increase the maximum profile length in the SAF Class/Type Descriptor table in z/OS.

`MAX-SAF-PROF-LENGTH=<nn>` Max resource profile length.

Password to Uppercase

To cater for situations where a site is in transition from uppercase to mixed case passwords setting this parameter can convert all passwords to uppercase. It is not recommended you use this option by default.

PASSWORD-TO-UPPER-CASE={NO,YES}	Convert password to uppercase.

Security Level

By default, EntireX Security furnishes authentication with optional encryption of send/receive buffers. The following parameter can be used to modify the functionality of EntireX Security:

SECURITY-LEVEL=ENCRYPTION	No authentication or authorization checks performed. The only functionality available in this mode is message privacy.
SECURITY-LEVEL=AUTHENTICATION	User authentication is performed but without any resource authorization (the normal default operation).
SECURITY-LEVEL=AUTHORIZATION	User authentication and resource authorization are both applied.

Caution:
In version 8.0, the default value for this parameter was "AUTHORIZATION"

Verified Client User ID

It is often important for server applications to know the identity of the client issuing the request. For this reason, the Broker kernel communicates the ACI field CLIENT-UID to the server application during the RECEIVE function. EntireX Security guarantees that the CLIENT-UID has been formally authenticated. EntireX Security automatically substitutes the value from trusted user ID where this is applicable.

PROPAGATE‑TRUSTED‑USERID=YES	Set ACI field CLIENT-UID to the user ID of the client. This will be authenticated by EntireX Security and may be obtained according to the trusted user ID mechanism where installed.
PROPAGATE‑TRUSTED‑USERID=NO	Do not set this value unless explicitly instructed to do so by Software AG support.

Security Node

This parameter can be used to specify a prefix which is added to all authorization checks, hence enabling broker kernels in different environments to perform authorization checks on different sets of resource profiles. For example, it is often important to distinguish among production, test, and development environments when performing authorization checks. The following settings are available:

SECURITY-NODE=YES	This causes the Broker ID - i.e., ETB113 - to be used as a prefix for all authorization checks.
SECURITY-NODE=<*node_name*>	This will utilize the string "node_name" (maximum 8 characters) as the prefix for all authorization checks.
SECURITY-NODE=NO	This causes the actual text (max 8 characters) to be prefixed onto all authorization checks..

Client RPC Authorization

For services supporting Natural RPC or other applications that know RPC, you can optionally perform authorization checks on the client making the RPC request by defining the "per service" attribute CLIENT-RPC-AUTHORIZATION=YES in the Broker attribute file. Setting this parameter to "YES" will cause the RPC library and program names to be appended to the profile associated with the authorization check. The resource profile would then appear as follows:

```
Class.server.service.rpc-library.rpc-program
```

Note:
Natural Security performs its resource authorization checks as follows:
<prefix-character>.rpc-library.rpc-program
To allow conformity with Natural Security, the CLIENT-RPC-AUTHORIZATION parameter can optionally be defined with a prefix character as follows:
CLIENT-RPC-AUTHORIZATION=(YES,<*prefix-character*>).

Considerations for Mainframe Natural Application Components

Application components running in a mainframe Natural environment which communicate using EntireX Broker interact with EntireX Security in the following ways:

- No password is required for applications executing under mainframe Natural where the trusted user ID option is implemented. This is true for both client and server application components. EntireX Security automatically acquires the logged-on user ID. Utilizing the trusted user ID avoids having to supply the password again. It also requires the customers to configure security for their mainframe environment(s), for example,ensuring that the CICS system is protected by RACF.

- Applications can override the trusted user ID by supplying a valid user ID/password combination in the ACI control block. This causes EntireX Security to ignore the trusted user ID in favor of the supplied credentials. Applications must therefore ensure that they do not assign an incorrect user ID or spurious password to the ACI control block, where trusted user ID is implemented. The CLIENT-ID as conveyed in the ACI to the server component of the application now represents the client’s verified user ID, derived either from valid user ID/password credentials or from trusted user ID itself.

Resource Profiles in EntireX Security

This section describes the definitions required in the SAF repository according to the underlying security system used (RACF, CA ACF2, CA Top Secret). It covers the following topics:

- Introduction
- Format of Resource Profiles
- Resource Definitions

Introduction

EntireX Security enables the secure deployment of EntireX Broker. This involves defining the resource profiles in the SAF repository to protect all distributed and mainframe application components. This philosophy is consistent with maintaining a single user ID and password.

Each SAF security system provides the facilities required for maintaining resource profiles.

RACF enables the grouping of similar resource profiles into a resource Class. CA ACF2 provides resource types which give equivalent functionality.

The name of the SAF class/type used to hold the EntireX-related resource profiles is specified with the Security-specific attribute `SAF-CLASS`. Default is `NBKSAG`.

The default length of the resource profile is 80 bytes, and this can be increased if necessary. See `MAX-SAF-PROF-LENGTH`. If you increase the maximum profile length, you must also increase the maximum profile length defined in the RACF class descriptor table.

Format of Resource Profiles

This section describes the format of various resource profiles. Note that the specific contents of resource files themselves will vary, depending upon the configuration options specified in the *Security-specific Attributes* under *Broker Attributes*.

EntireX Broker

- **Client Server Example**
  Resource profiles protecting Broker client and server applications normally comprise Broker class, name and service. It is possible to omit any of these components from the resource profile. See also *Request Authorization*. The following resource profile shows an example service:

  ```
  ETB.POLICY.QUOTE1
  ```

  Client applications must execute with a user ID that has READ access to allow them to send to the given service. Registration of services is also secured. Server applications require CONTROL access to register a service with the Broker.

- **Publis and Subscribe Example**
  Resource profiles protecting Broker publish and subscribe applications are always defined in terms of the 96 character topic name. The following resource profile would be used to protect a topic used for publish and subscribe:

```
NYSE
```

Subscriber applications must execute with a user ID that has READ access to this resource to allow them to issue the subscribe command for this topic. Publisher applications require CONTROL access in order to send publications to the topic.

EntireX Broker TCP/IP Address Verification

If optional TCP/IP address checking is required at authentication time, the relevant resource profiles must be defined in the security system. Users will require READ access in order to connect, using TCP/IP, from a particular address. A typical TCP/IP address would be entered in the security system as follows:

```
247.72.46.239
```

Note:
You can perform TCP/IP address checking independently of user authentication and authorization, by setting `CHECK-IP-ADDRESS=YES` and `SECURITY-LEVEL=ENCRYPTION`, i.e. not authentication or authorization. This results in an authorization check for the IP address for the user ID under which EntireX Broker itself executes.

Command and Information Services

Access to Command and Information Services is controlled by permitting, or denying, access to Software AG supplied services which implement Command and Information Services.

For a complete list of profiles representing these services, see *Authorization for Command and Information Services*.

For more information see *Security with Command and Information Services*.

Resource Definitions

This section describes the definitions required in the various supported security systems in order to enable Security for EntireX resources. These definitions are described in the following subsections:

- Defining Resources to RACF
- Defining Resources to CA Top Secret
- Defining Resources to CA ACF2

Note:
Define resources using uppercase characters only.

Defining Resources to RACF

This section defines how the EntireX resources are defined to RACF. For exact details of the procedures to be followed for the installed RACF version, consult the relevant IBM manuals.

Overview of tasks:

- Add classes to class descriptor table

- Update z/OS router table
- Activate new classes
- Assign user ID for the Broker started task, if you have not done so already
- Permit user access to resource profiles
- Optimize the performance of RAC authorization checks

To add classes to class descriptor table

1. Add the resource classes to the RACF class descriptor table. Refer to the IBM SPL RACF manual.

   For an example, see IBM `SYS1.SAMPLIB`, member `RACINSTL`.

2. You must allocate a class descriptor length for class NBKSAG of 80 bytes in order to prevent the possibility of a system 282 abend, which could occur if the length of your resource (class/server/service) exceeds the length known to RACF. The maximum length allowed by EntireX is 80 bytes, so allow 80 bytes in the RACF class descriptor table.

3. Define the classes to enable discrete and generic profile use.

4. Check further attributes controlling the level of RACF messages generated when performing `RACROUTE` calls, as well as the required level of SMF recording. Sample definitions are provided in source member `RACFCLSX`.

To update z/OS router table

- Update the z/OS router table as described in the IBM SPL RACF manual. For an example, see the IBM `SYS1.SAMPLIB`, member `RACINSTL`, section `RFTABLE`.

To activate new classes

- Activate new resource classes with `SETROPTS` (see *IBM RACF Command Language Reference manual*). For an example, activate class `NBKSAG`:

```
SETROPTS CLASSACT(NBKSAG)
SETROPTS GENCMD(NBKSAG)
SETROPTS GENERIC(NBKSAG)
```

To assign user ID for the Broker started task

- The EntireX Security functions are performed within the address space of EntireX Broker. Assign a user ID to the Broker started task with the relevant RACF authorizations, including the ability to perform `RACROUTE, TYPE=EXTRACT`, `TYPE=AUTH` and `TYPE=VERIFY` calls on profiles belonging to the defined classes.

To permit user access to resource profiles

- After adding profiles to protect the different resources, permits users the required level of access, using the relevant RACF commands. The following example adds resource profile `ETB.POLICY.QUOTE1` and grants read access to user ID USER2 and control access to USER3. USER2 represents a client and requires read access to execute while USER3 represents a server component which needs control access to register:

```
RDEFINE NBKSAG ETB.POLICY.QUOTE1 UACC(NONE)
PERMIT ETB.POLICY.QUOTE1 CLASS(NBKSAG) ACCESS(READ) ID(USER2)
PERMIT ETB.POLICY.QUOTE1 CLASS(NBKSAG) ACCESS(CONTROL) ID(USER3)
```

- If you utilize authorization checks based upon TCP/IP address (TCP transport only) then define these resource definitions (`RDEFINE`) as follows and `PERMIT` the appropriate user read access as shown:

```
RDEFINE NBKSAG 247.72.46.239 UACC(NONE)
PERMIT 247.72.46.239 CLASS(NBKSAG) ACCESS(READ) ID(USER42)
```

To optimize the performance of RACF authorization checks

- Use `SETROPTS RACLIST(NBKSAG)` to cache in memory the RACF general resource profiles belonging to class `NBKSAG`. If you use a RACF resource class other than `NBKSAG`, make sure this RACF general resource class is cached in memory.

Defining Resources to CA Top Secret

This section defines how the EntireX classes are defined to CA Top Secret. For exact details of the procedures to be followed for the installed version of CA Top Secret, consult the relevant CA Top Secret manual.

Overview of tasks:

- Add CA Top Secret Facility
- Assign user ID for the Broker started task, if you have not done so already
- Add procedure name for the started task
- Add resource type to resource definition table
- Assign ownership of resources
- Permit defined resources to users

To add CA Top Secret Facility

- CA Top Secret enables a set of authorization checks to be made against a certain facility. For example, this can be used to secure the development environment `SAGDEV` separately from the production environment `SAGPROD`. Alternatively, a default facility of batch can be used.

  To add additional facilities, use the following commands:

```
AUTHINIT,MULTIUSER,NONPWR,PGM=ETBNUC,NOABEND
```

To assign a user ID for the Broker started task

- Add one user ID for each instance of the Broker started task.

  If required, different facilities can be assigned to development and production started tasks.

  The designated facility is assigned to the started task user ID:

```
TSS CRE(user-id) DEPT(dept) MASTFAC(fac)
```

To add a procedure name for the Broker started task

- The procedure name under which the Broker started task executes must be defined to CA Top Secret.

```
TSS ADD(STC) PROC(proc) ACID(user-id)
```

To add resource type to resource definition table

- Add the resource types to the CA Top Secret resource definition table (RDT). Resource definitions relating to EntireX are kept in resource type NBKSAG. Refer to the *CA Top Secret Reference Guide* for a detailed explanation of the following commands and arguments:

```
TSS ADD(RDT) RESCLASS(NBKSAG)
RESCODE(HEXCODE)
ATTR(LONG)
ACLST(NONE,READ,CONTROL)
DEFACC(NONE)
```

To assign ownership of resources

- Assign ownership to a particular resource as shown in the following example. This must be done before permitting access to defined resource profiles:

```
TSS ADD(user1) NBKSAG(etb.policy.quote1)
```

This makes user *user1* the owner of the Broker service *etb.policy.quote1*.

Similarly, to add ownership to profiles used to control access based on TCP/IP address (TCP communications only) follow the steps below. This makes *user4* the owner of this resource profile:

```
TSS ADD(user4) NBKSAG(247.72.46.239)
```

To permit defined resource to users

- Permit access to a resource profile as in the following example. In the example, user *user2* is permitted read access to the Broker service *etb.policy.quote1*. This enables the user to execute as a client and issue requests to this Broker service:

```
TSS PER(user2) NBKSAG(etb.policy.quote1) FAC(fac) ACCESS(READ)
```

Similarly, to permit access to profiles used to control access based on TCP/IP address, use the PER command as shown:

```
TSS PER(user42) NBKSAG(247.72.46.239) FAC(fac) ACCESS(READ)
```

Defining Resources to CA ACF2

See also your CA ACF2 documentation.

Note:
CA ACF2 provides insufficient return codes to determine whether a resource profile does not exist or simply the user does not have access to it. Therefore, if access is denied by CA ACF2, EntireX Security will always report "Access denied resource not allowed" in the error message.

To define resources to CA ACF2

1. The Broker or Broker Services started task executes as a normal started task in z/OS. Define the user ID of started task to CA ACF2 with the following attributes:

   ```
   MUSASS, STC
   ```

2. Insert SAFDEF records as follows:

   ```
   SAFDEF.EXS1
   FUNCRET(4) FUNCRSN(0) ID(ENTIREX) MODE(GLOBAL)
   RACROUTE(REQUEST=VERIFY SUBSYS=ETBNUC REQSTOR=-)
   RETCODE(4)

   SAFDEF.EXS2
   FUNCRET(4) FUNCRSN(0) ID(ENTIREX) MODE(GLOBAL)
   RACROUTE(REQUEST=AUTH SUBSYS=ETBNUC REQSTOR=-)
   RETCODE(4)

   SAFDEF.EXS3
   FUNCRET(4) FUNCRSN(0) ID(ENTIREX) MODE(GLOBAL)
   RACROUTE(REQUEST=EXTRACT SUBSYS=ETBNUC REQSTOR=-)
   RETCODE(4)
   ```

3. For the general resource class name used by EntireX Security, define a 3-character CA ACF2 resource type code by inserting a `CLASMAP` record as follows:

   ```
   CLASMAP
   ENTITYLN(0) MUSID() RESOURCE(NBKSAG) RSRCTYPE(NBK)
   ```

4. Define the required security profiles to CA ACF2 using the new type code.

   The following example shows the addition of a Broker service *etb.policy.quote1*, allowing read access only for user ID *user2*:

   ```
   $KEY(ETB) TYPE(NBK)
   policy.quote1 UID(user2) SERVICE(READ)   ALLOW
   policy.quote1 UID(-)                     PREVENT
   ```

   A service level of DELETE is required for a service to register (this is functionally the same as CONTROL access in RACF).

   The following example secures the TCP/IP connection for checking at authentication time for the TCP/IP address of 247.72.46.239 granting access to all users, except U402451.

   ```
   $KEY(247) TYPE(NBK)
   72.46.239 UID(user42)  SERVICE(READ) ALLOW
   72.46.239 UID(u402451) SERVICE(READ) ALLOW
   ```