webMethods EntireX

EntireX Security

Version 10.8

October 2022

WEBMETHODS

This document applies to webMethods EntireX Version 10.8 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1997-2022 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at http://softwareag.com/licenses.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at http://softwareag.com/licenses/ and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at http://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: EXX-SECURITY-108-20220601

Table of Contents

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

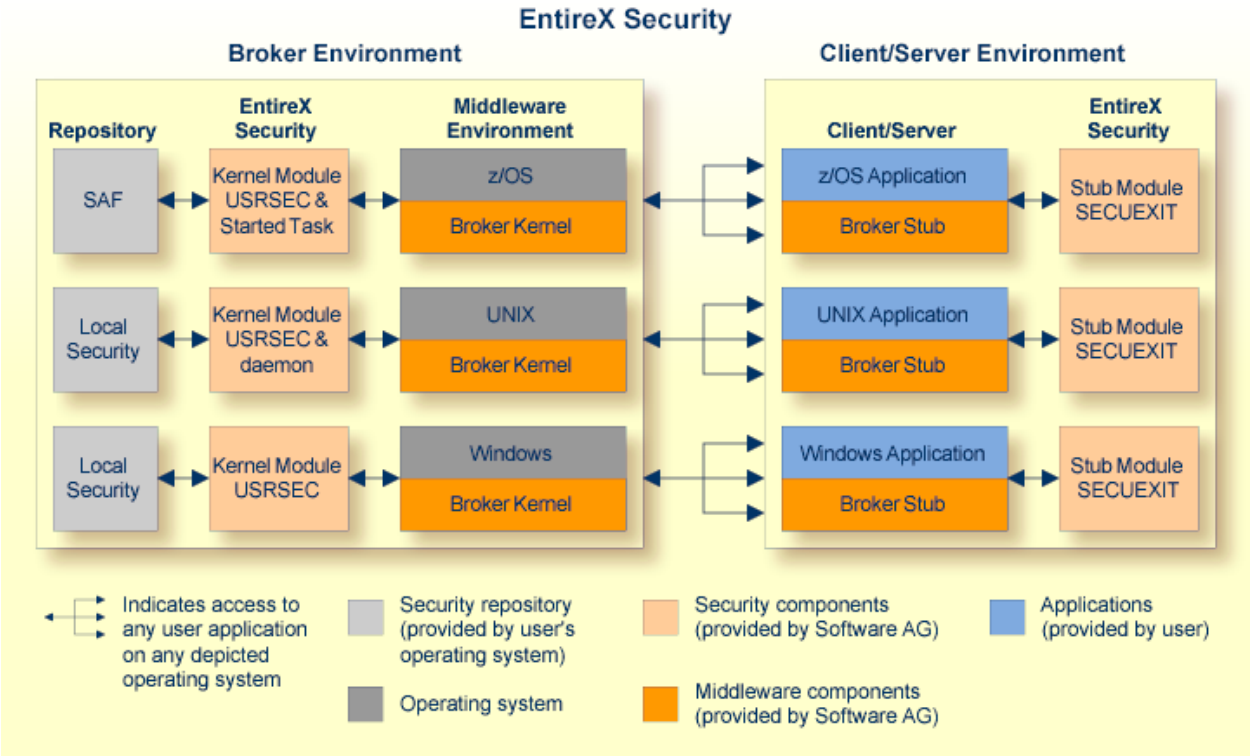
2 Introduction to EntireX Security

EntireX Security is the standard security solution provided with EntireX. It provides centralized security for EntireX Broker under z/OS, UNIX and Windows. EntireX Security operates with your organization's security repository.

Note: Trace files can contain sensitive personal data (user ID, IP address, SSL certificates and payload data). This is particularly relevant if you have activated EntireX Security. EntireX uses trace files for accounting, diagnostics and error analysis. We recommend you check the different trace opportunities provided by EntireX and delete trace files if they are no longer needed. The various EntireX components will not delete these trace files automatically; this is your responsibility as user. Use the appropriate tools of the respective operating system.

Location of Security Components

This diagram shows the locations where the broker kernel and broker stubs can be installed; it also shows the locations of the security components of the kernel and stubs.



The *Platform Coverage* shows where broker kernel and stubs are supported.

Overview of EntireX Security Features

EntireX Security provides comprehensive security for EntireX Broker:

- user authentication
- user authorization
- supplied in object code only

The major advantages of EntireX Security:

- **Comprehensive Security**
  EntireX Security provides comprehensive security for EntireX Broker, that is user authentication and user authorization
- **Protection of Application Systems**
  EntireX Security protects client and server application systems.
- **No User Exits to Write/Debug**
  EntireX Security is fully supported (that is, object code only). There are no user exits to write and debug. In most installations EntireX Security operates without altering runtime applications.
- **One User=One Definition**
  EntireX Security allows your organization to control the use of all applications, including distributed components, from a central point, enabling flexible control with a "one user = one definition" approach.
- **Standard Security Definitions**
  EntireX Security enables security definitions, based on class/name/service (client and server), to be validated. All definitions are managed using existing security procedures and software.
- **Protected Investment in SAF-based Security Repositories**
  On z/OS security definitions are accessed using industry standard SAF interface. Your investment in SAF-based security repositories is therefore protected. This includes not only the security systems RACF, CA ACF2 and CA Top Secret, but also the infrastructure to administer security profiles.

Functionality of EntireX Security

This section covers the following topics:

Authentication of User

Authentication verifies whether the identity specified by the user application is the actual identity. Authentication is performed for application components executing on different platforms against the security repository where the broker kernel resides. It is the responsibility of the application to supply the user ID and password.

Authorization of Client and Server

Authorization determines whether client and server application components are allowed to execute with EntireX Broker. The class, server and service associated with the user's command form the basis for the check. Separate authorization checks are performed, depending on the role of the application as either client or server. The checks differentiate between the client's `SEND` command and a server's `REGISTER` command. Therefore your security administrator should allow only the level of access required for the user to operate in the intended role. The authorization checks are performed on the same platform as the broker kernel resides (see ***Location of Security Components***) regardless of location of the individual application components.

This authorization functionality is available only with EntireX Broker running under z/OS. Under UNIX and Windows, limited functionality is available through authorization rules. See also *Authorization Rules*.

Authorization for Command and Information Services

Authorization determines whether a user is permitted to issue commands to the EntireX Broker Command and Information Services. See *Broker Command and Information Services*. The following resource definitions, derived from the user's intended activities, form the basis for the check. The level of authorization needed for accessing these services is identical to that of a "client". These services are automatically started by broker kernel without performing a check for `REGISTER`:

Resource Definition	Using
SAG.ETBCIS.CMD	ETBCMD
SAG.ETBCIS.INFO	ETBINFO to retrieve general information. Specify INFO for the full information service: all clients, servers and conversations are listed.
SAG.ETBCIS.SAGCCV5	For RPC CIS command services.
SAG.ETBCIS.SAGCIV5	For RPC CIS information services.
SAG.ETBCIS.SECURITY-CMD	For security related requests: (1) reset user [ACEE]; (2) change security trace level.
SAG.ETBCIS.USER-INFO	ETBINFO to retrieve information specific to the user issuing the command. USER-INFO is an information service limited to user-specific information: only the user's own resources are listed.

In addition, a separate authorization check is made when a user attempts to perform third party actions affecting other users:

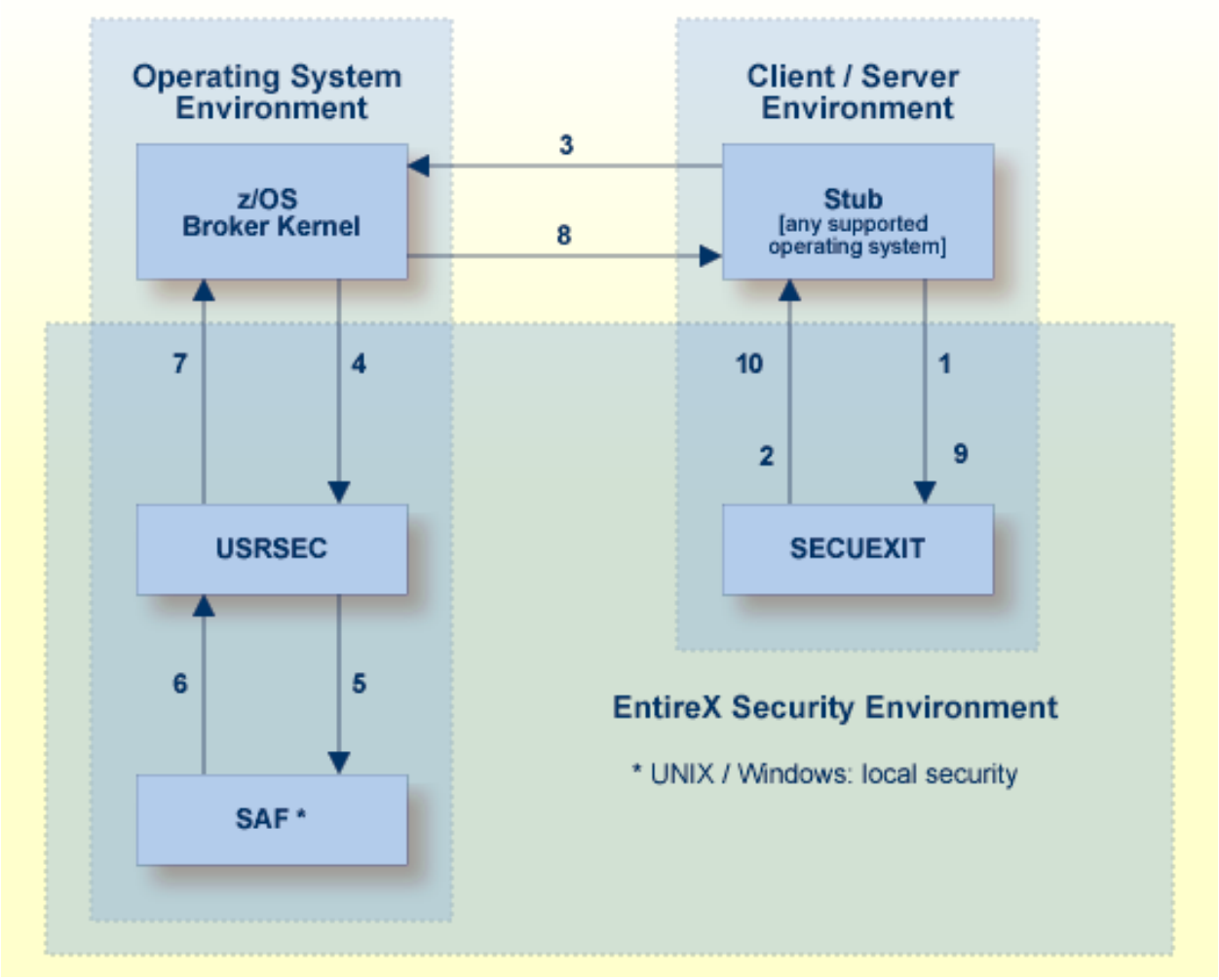
- To shut down a *service*, users must have the required authorization to register this class, server and service themselves.
- To shut down a *server*, users must have the required authorization to register all the services registered by that server.

This authorization is required in addition to the requesting user's ability to use SAG.ETBCIS.CMD in general.

This authorization functionality is available only with EntireX Broker running under z/OS. Under UNIX and Windows, limited functionality is available through authorization rules. See also *Authorization Rules*.

Data Flow of EntireX Security (Client and Server)

The diagram shows the location of the security components of the kernel and stubs of EntireX Broker. Each step in the table below represents a specific step in the data flow sequence.This table describes the functionality of the security components of the kernel / stubs of broker: authorization and authentication.



Note: This diagram depicts the operation of the broker stub for Natural and other third-generation programming languages. It is not intended to show the mechanism used by the Java ACI and *EntireX Adapter* with regard to EntireX Security. See *Using EntireX Security with Java-based EntireX Applications* under *Writing Advanced Applications - EntireX Java ACI*.

Description of Steps in Data Flow

1. Broker stub calls security module `SECUEXIT`, if present.
2. Security module `SECUEXIT` encrypts the password.
3. Broker stub communicates the call to the broker kernel.
4. Broker kernel calls security module `USRSEC`, which provides the functionality, based on the *Configuration Options for Broker*:
   - re-authentication if a user acquires a new physical user ID
   - re-authentication if the value of a user's ACI security token changes.

   All functionality is available on z/OS only.
5. Security module `USRSEC` references local security system where the broker is located:
   - **z/OS**
     Security module `USRSEC` calls SAF (RACF, CA ACF2 or CA Top Secret).
   - **UNIX**
     Security module `USRSEC` calls the UNIX security system or LDAP.
   - **Windows**
     Security module `USRSEC` calls the Windows security system or LDAP.
6. The result of the security check is communicated back to security module `USRSEC`.
7. Security module `USRSEC` passes call to Broker kernel.
8. Broker kernel communicates the call to Broker stub of the partner application.
9. The Broker stub calls `SECUEXIT`.
10. Security module `SECUEXIT` returns call to Broker stub.

For installation see *Installing/Setting up EntireX Security* under z/OS | UNIX | Windows | BS2000 in the platform-specific Installation documentation.

Glossary of Terms

See also *EntireX Glossary*.

Authentication

Authentication verifies whether the identity specified by the user ID in the ACI control block is the actual identity. Authentication is performed by checking the user's ID and password against a security system, except where Trusted user ID automatically acquires the identity of the logged-on user or batch job, obviating the requirement for a password in the ACI control block. See ***Trusted User ID***. Trusted user ID is applicable only where the application component and the broker kernel reside under z/OS.

Authentication is not performed with every call. It is performed when a user is first presented to the kernel of EntireX Broker. The broker kernel recognizes the identity of the user on subsequent occasions by combination of user ID and physical user ID (or user ID and token where supplied). Broker kernel also verifies the correctness of the ACI security token on all subsequent commands and, if this is not as expected, the application must provide the correct user ID and password again (unless configured otherwise).

An application identifying itself by combination of user ID and token can change its physical user ID without needing to provide the user ID and password again provided it maintains the value of ACI security token in the broker control block. This functionality is recommended for multithreading applications or applications executing within a Web server. Caution should be exercised to ensure the user ID and token combination is unique.

Authorization

Authorization is performed when:

- a client issues a request to a service in the case of the first `SEND` command in a conversation, or of each `SEND` command if `CONV-ID=NONE`
- a server registers a service to the broker
- an application connects to broker through TCP/IP, an optional authorization check is performed based on the address

Full authorization functionality is available only under z/OS.

Broker Kernel

It is the location of the broker kernel that determines the point at which the authentication and authorization checks are performed. ***Authentication*** and ***Authorization*** are performed in the kernel.

See *Platform Coverage* for where EntireX Broker kernel is supported.

Broker Stub

In EntireX Broker, a module that implements the ACI (Advanced Communication Interface) is commonly referred to as “broker stub” or simply “stub”. Stubs are installed on the client side or server side.

See *Platform Coverage* for where broker stubs are supported.

3 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.

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

Functionality of EntireX Security

This table lists the security functionality available with EntireX Security running broker kernel under the respective operating system.

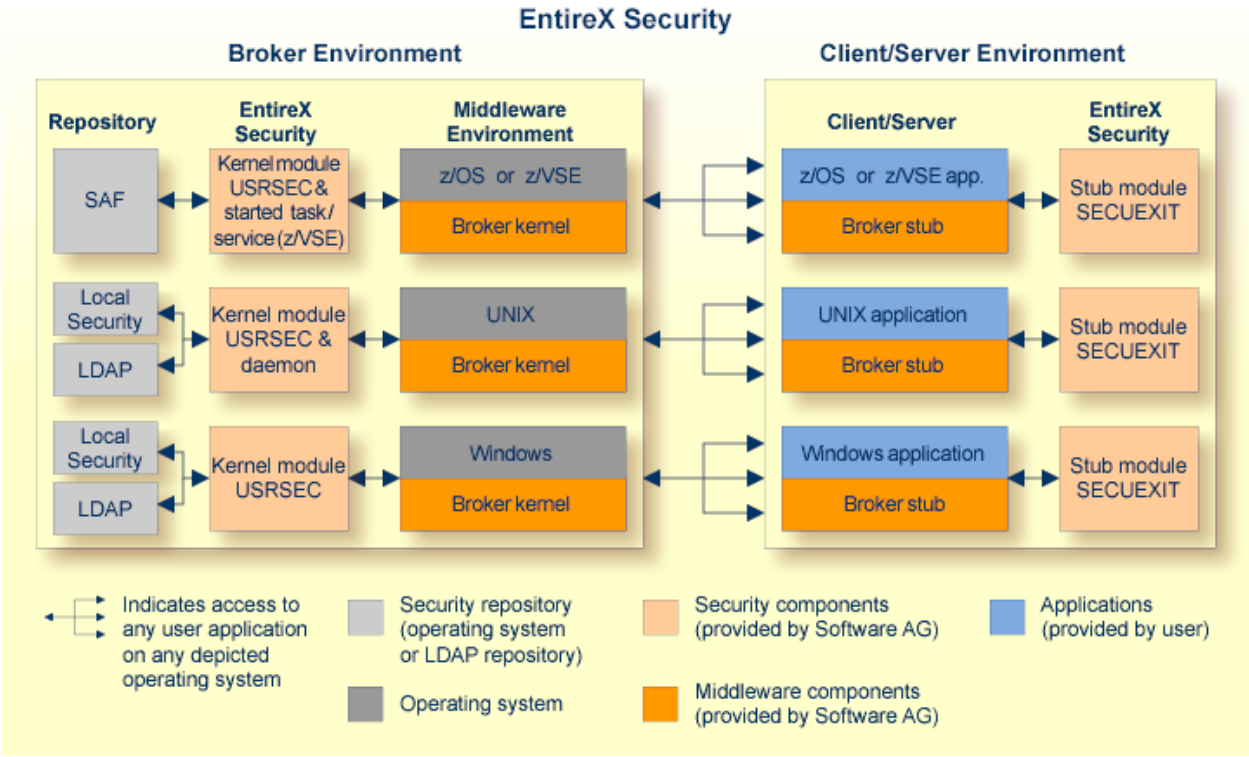
Security Functionality	z/OS	UNIX	Windows	BS2000	Comment
Authentication of user	Yes	Yes	Yes	Yes	Verify broker user ID and broker password sent by an application to the broker. Under z/OS the broker verifies a long broker password as a RACF password phrase.
User password change	Yes	No	No	No	
LDAP authentication	No	Yes	Yes	No	Authenticate using LDAP repository.
Trusted user ID	Yes	No	No	No	Trusted computing base, avoiding plain text password.
Verified client user ID	Yes	No	No	Yes	Provide verified identity of client to server.
Authorization of client request	Yes	No	No	No	
Authorization of server register	Yes	No	No	No	
Authorize IP connection	Yes	No	No	No	
Authorization rules	No	Yes	Yes	No	An authorization rule is used to perform access checks for authenticated user IDs against lists of services defined within the rule. This feature is available on UNIX and Windows using EntireX Security on these platforms. Authorization rules can be stored in the Broker attribute file or in an LDAP repository. See *Authorization Rules*.
SSL/TLS	(1)	Yes	Yes	No	Industry standard encryption mechanism. See *SSL/TLS, HTTP(S), and Certificates with EntireX*.
Using SSL Certificates for Authentication	Yes	No	No	No	See ***Using SSL Certificates for Authentication***.

Notes:

1. Establishing an SSL/TLS connection on z/OS is done with IBM's Application Transparent Transport Layer Security (AT-TLS)

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.



The 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.

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*.

- Authentication
- Trusted User ID

- Request Authorization
- Request Authorization for Command and Info Services
- Ignore Security Token
- 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

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

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.

An *Implicit Logon* is possible where the attribute file contains `AUTOLOGON=YES`. This means the first command you issue does not have to be `LOGON`. In this case the application must supply user ID and password credentials for commands `SEND` or `REGISTER`.

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.

For more information see *Authentication* under *Writing Applications using EntireX Security* in the ACI Programming documentation.

Trusted User ID

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 must not provide a password if they intend to use trusted user ID, see *Enable Trusted User ID (Optional)* under *Installing EntireX Security under z/OS*. Activate this option with Security-specific broker attribute `TRUSTED-USERID`:

`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.

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`.

- RPC Server for CICS ECI
- RPC Server for IMS Connect
- RPC-ACI Bridge
- RPC Server for IBM MQ
- RPC Server for Java
- RPC Server for XML/SOAP

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.

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

The following parameters can be used to modify the functionality of EntireX Security:

`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.

Note: 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.

Note: There is an uppercase translation when the `USER-ID` field is propagated to the `CLIENT-UID` field under EntireX Security when the broker kernel is running under z/OS.

`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 - that is, 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).

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*.

EntireX Broker Client/Server

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.

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 by setting `CHECK-IP-ADDRESS=YES`. 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(u402451) SERVICE(READ) PREVENT
72.46.239 UID(-) SERVICE(READ) ALLOW
```

Using SSL Certificates for Authentication

With EntireX 10.7 and above, Broker on z/OS supports authentication of participants with their SSL certificate. The following configuration steps are required to enable this feature:

- Step 1: Create Certificates
- Step 2: Define Mapping Rules in RACF to Authenticate with Certificates
- Step 3: Add Connection in AT-TLS-controlling Mode to the Policy Agent
- Step 4: Configure Authentication with Certificates in Broker
- Step 5: Run SSL Clients

Step 1: Create Certificates

Creating certificates is described under *Creating Certificates with OpenSSL (z/OS, UNIX, Windows)* in the platform-independent Administration documentation.

Step 2: Define Mapping Rules in RACF to Authenticate with Certificates

The created certificates contain information about Issuer and Subject. The IBM documentation ***z/OS Security Server RACF Security Administrator's Guide*** provides an example of how to use Issuer data for a mapping to IBM user IDs. The following example illustrates a sample filter based on the certificates of EntireX with changes to authenticate `EXXOKAY`.

```
RACDCERT MULTIID MAP WITHLABEL('CertGroup') TRUST
         IDNFILTER('CN=DefaultCA.OU=EntireX.O=Software AG.L=Darmstadt.C=DE')
         SDNFILTER('CN=ibm2.OU=EXXOKAY.O=Software AG.L=Darmstadt.C=DE')
         CRITERIA(APPLID=&APPLID)
SETROPTS RACLIST(DIGTNMAP) REFRESH

RDEFINE  DIGTCRIT APPLID=BROKER APPLDATA('USER1')
RDEFINE  DIGTCRIT APPLID=SERVER APPLDATA('USER2')
SETROPTS RACLIST(DIGTCRIT) REFRESH
```

Step 3: Add Connection in AT-TLS-controlling Mode to the Policy Agent

The following is an excerpt of a Policy Agent, MVS task PAGENT configuration file defining a Broker TCP/IP port as secure SSL port in AT-TLS-controlling mode as outcome of this configuration:

```
TTLSRule                          ConnRule01~34
{
  LocalAddr                       ALL
  RemoteAddr                      ALL
  LocalPortRangeRef               portR19
  RemotePortRangeRef              portR2
  Jobname                         <job_name> e.g. ETBNUC
  Direction                       Inbound
  Priority                        222
  TTLSConnectionActionRef         cAct11
  TTLSEnvironmentActionRef        eAct9
  TTLSGroupActionRef              gAct1
}
TTLSConnectionAction              cAct11
{
  HandshakeRole                   ServerWithClientAuth
  TTLSCipherParmsRef              cipher1
  TTLSConnectionAdvancedParmsRef  cAdv10
  CtraceClearText                 Off
  Trace                           6
}
TTLSEnvironmentAction             eAct9
{
  HandshakeRole                   ServerWithClientAuth
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR1
}
TTLSGroupAction                   gAct1
{
  TTLSEnabled                     On
  Trace                           6
}
TTLSConnectionAdvancedParms       cAdv10
{
  ApplicationControlled           On
  CertificateLabel                <certificate_label> e.g. ExxAppCert
```

```
  SecondaryMap                    Off
  SSLv3                           Off
  TLSv1                           On
  TLSv1.1                         On
  TLSv1.2                         On
}
TTLSKeyringParms                  keyR1
{
  Keyring                         <user_id / keyring> e.g. EXX/EXXRING
}
PortRange                         portR19
{
  Port                            <port_number> e.g. 1958
}
PortRange                         portR2
{
  Port                            1024-65535
}
CipherParms                       cipher1
{
  V3CipherSuites                  TLS_RSA_WITH_AES_256_GCM_SHA384
  ...
  V3CipherSuites                  TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
}
```

The sample configuration under *Migration from Broker's Direct SSL/TLS Support to AT-TLS* differs from the configuration listed above in parameters `HandshakeRole` and `ApplicationControlled` (see *red* lines). These parameters are needed to establish client authentication and an AT-TLS-controlling application.

Step 4: Configure Authentication with Certificates in Broker

Using SSL certificates for authentication requires a separate TCP/IP port in Broker. This port must be configured as AT-TLS-controlling connection and cannot be used for connections without certificate-based authentication. The following Broker attributes are required:

```
DEFAULTS = BROKER
  SECURITY = YES

DEFAULTS = SECURITY
  APPLICATION-NAME = <applid> e.g. BROKER

DEFAULTS = TCP
  PORT = <port> e.g. 3932
  CERT-AUTHENTICATION = YES
```

You may define up to five TCP/IP ports with attribute `CERT-AUTHENTICATION` in the `DEFAULTS = TCP` section. See sample section below:

```
DEFAULTS = TCP
  PORT = 3930 , CERT-AUTHENTICATION = NO  *** TCP/IP
  PORT = 3931 , CERT-AUTHENTICATION = NO  *** AT-TLS-unaware
  PORT = 3932 , CERT-AUTHENTICATION = YES *** AT-TLS-controlling
  PORT = 3933 , CERT-AUTHENTICATION = NO  *** AT-TLS-unaware
  PORT = 3934 , CERT-AUTHENTICATION = NO  *** TCP/IP
```

Step 5: Run SSL Clients

Before running any SSL clients, make sure the broker is prepared for certificate-based authentication. Value 3932 is used as port in our sample configuration, see ***Step 4: Configure Authentication with Certificates in Broker***.

- UNIX and Windows
- z/OS

UNIX and Windows

EntireX client and server programs are SSL clients and specify the following parameters in the send buffer of ACI function `SETSSLPARMS` to communicate with the Broker:

- `key_file`
- `key_passwd`
- `key_store`
- `trust_store`
- `verify_server`

To make sure that each SSL participant (thread) presents a valid certificate for authentication, the ACI function `SETSSLPARMS` needs to be performed for each thread to create an SSL connection to the broker. This is only relevant if you are using native ACI programming in combination with threads. Also note that when you use SSL transport, socket pooling is ignored (see `ETB_SOCKETPOOL` under *Environment Variables in EntireX*). See also *SSL/TLS Parameters for SSL Clients* under *SSL/TLS, HTTP(S), and Certificates with EntireX* in the platform-independent Administration documentation.

To test certificate-based authentication

- Use client program `bcoc` and server program `bcos`.

```
bcos ↩
"-xVERIFY_SERVER=N&TRUST_STORE=ExxCACert.pem&KEY_STORE=ExxAppCert.pem&KEY_FILE=ExxAppKey.pem&KEY_PASSWD=ExxAppKey" ↩
"-i1" "-b<host>:<port>:SSL"

bcoc ↩
"-xVERIFY_SERVER=N&TRUST_STORE=ExxCACert.pem&KEY_STORE=ExxAppCert.pem&KEY_FILE=ExxAppKey.pem&KEY_PASSWD=ExxAppKey" ↩
"-i1" "-b<host>:<port>:SSL"
```

Value 3932 is used as port in our sample configuration.

z/OS

Client and server programs you want to run on z/OS have to be added to the configuration of the Policy Agent. These programs use `<host>:<port>:TCP` as `BROKER-ID`.

Achieving FIPS Compliance

The z/OS operating system can be configured to run EntireX Broker FIPS 140-2 compliant. This requires changes in the following system components:

- AT-TLS (Application Transparent Transport Layer Security)
- System SSL
- ICSF (z/OS Integrated Cryptographic Service Facility)

The TCP/IP port of Broker must run under the control of AT-TLS, and all applications using Broker need the Secure Socket Layer protocol (TLS/SSL) as transport. You do not need to change the attribute file of Broker.

The IBM support pages provide the document *Setting up AT-TLS for FIPS 140 mode.pdf*. It contains an overview of necessary system changes and describes the steps in detail.

4 EntireX Security under UNIX

This chapter introduces EntireX Security under UNIX 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.

Note: Setting up EntireX Security is described under *Setting up EntireX Security under UNIX*.

Functionality of EntireX Security

This table lists the security functionality available with EntireX Security running Broker Kernel under the respective operating system.

Security Functionality	z/OS	UNIX	Windows	BS2000	Comment
Authentication of user	Yes	Yes	Yes	Yes	Verify broker user ID and broker password sent by an application to the broker. Under z/OS the broker verifies a long broker password as a RACF password phrase.
User password change	Yes	No	No	No	
LDAP authentication	No	Yes	Yes	No	Authenticate using LDAP repository.
Trusted user ID	Yes	No	No	No	Trusted computing base, avoiding plain text password.
Verified client user ID	Yes	No	No	Yes	Provide verified identity of client to server.
Authorization of client request	Yes	No	No	No	
Authorization of server register	Yes	No	No	No	
Authorize IP connection	Yes	No	No	No	
Authorization rules	No	Yes	Yes	No	An authorization rule is used to perform access checks for authenticated user IDs against lists of services defined within the rule. This feature is available on UNIX and Windows using EntireX Security on these platforms. Authorization rules can be stored in the Broker attribute file or in an LDAP repository. See *Authorization Rules*.
SSL/TLS	(1)	Yes	Yes	No	Industry standard encryption mechanism. See *SSL/TLS, HTTP(S), and Certificates with EntireX*.
Using SSL Certificates for Authentication	Yes	No	No	No	See ***Using SSL Certificates for Authentication***.

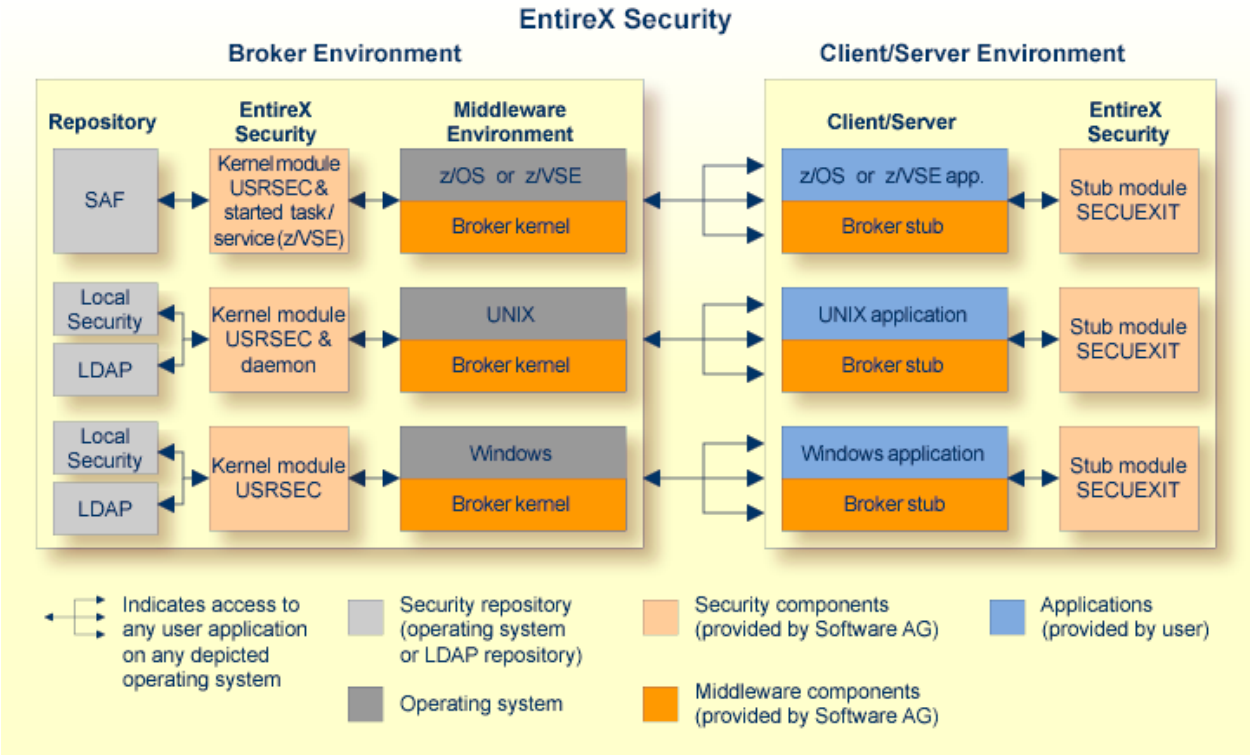
Notes:

1. Establishing an SSL/TLS connection on z/OS is done with IBM's Application Transparent Transport Layer Security (AT-TLS)

The 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.

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.



5 EntireX Security under Windows

This chapter introduces EntireX Security under Windows through overviews of the functionality and components of EntireX Security. The location where the broker kernel is installed determines the functionality made available for EntireX Security.

Note: Installation of the security software is described under *Setting up EntireX Security under Windows*.

Functionality of EntireX Security

This table lists the security functionality available with EntireX Security running Broker Kernel under the respective operating system.

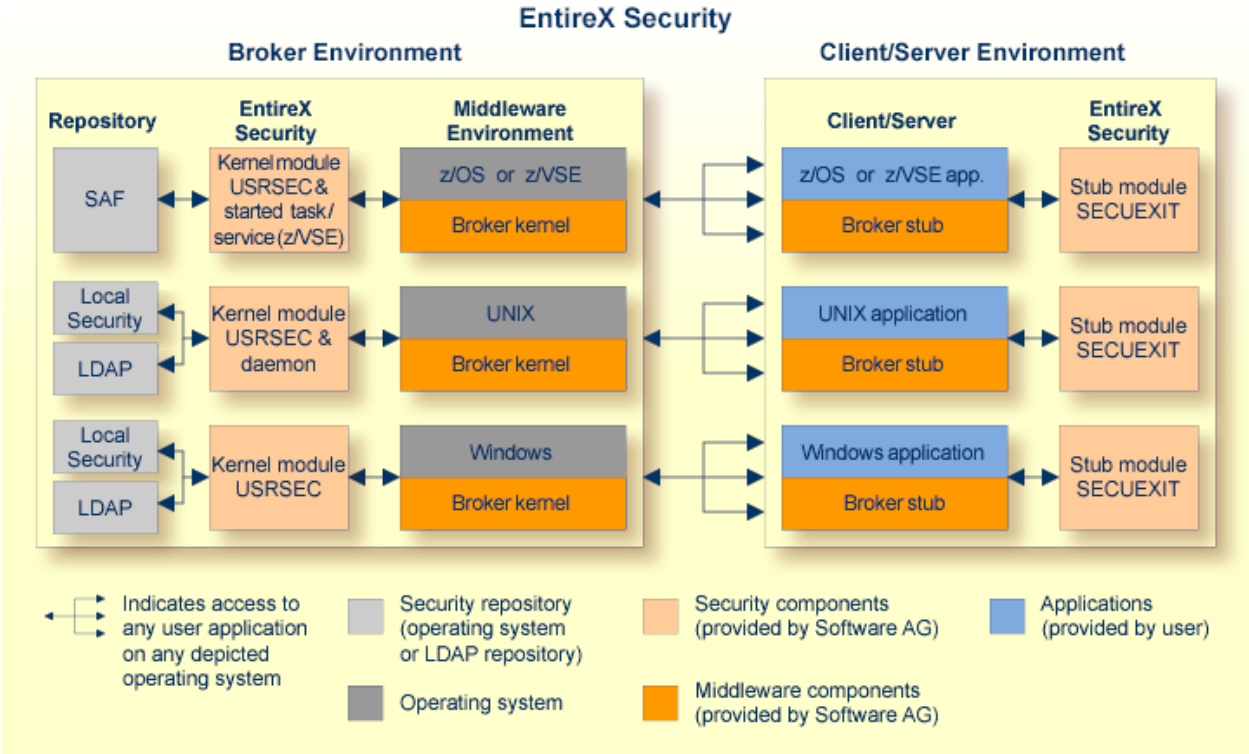
Security Functionality	z/OS	UNIX	Windows	BS2000	Comment
Authentication of user	Yes	Yes	Yes	Yes	Verify broker user ID and broker password sent by an application to the broker. Under z/OS the broker verifies a long broker password as a RACF password phrase.
User password change	Yes	No	No	No	
LDAP authentication	No	Yes	Yes	No	Authenticate using LDAP repository.
Trusted user ID	Yes	No	No	No	Trusted computing base, avoiding plain text password.
Verified client user ID	Yes	No	No	Yes	Provide verified identity of client to server.
Authorization of client request	Yes	No	No	No	
Authorization of server register	Yes	No	No	No	
Authorize IP connection	Yes	No	No	No	
Authorization rules	No	Yes	Yes	No	An authorization rule is used to perform access checks for authenticated user IDs against lists of services defined within the rule. This feature is available on UNIX and Windows using EntireX Security on these platforms. Authorization rules can be stored in the Broker attribute file or in an LDAP repository. See *Authorization Rules*.
SSL/TLS	(1)	Yes	Yes	No	Industry standard encryption mechanism. See *SSL/TLS, HTTP(S), and Certificates with EntireX*.
Using SSL Certificates for Authentication	Yes	No	No	No	See ***Using SSL Certificates for Authentication***.

Notes:

1. Establishing an SSL/TLS connection on z/OS is done with IBM's Application Transparent Transport Layer Security (AT-TLS)

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.



The 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.