Overview of 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.

This chapter covers the following topics:

- Introduction to EntireX Security
- Overview of EntireX Security Features
- Functionality of EntireX Security
- Data Flow of EntireX Security (Client and Server)
- Data Flow of EntireX Security (Publish and Subscribe)
- Glossary of Terms

Introduction to EntireX Security

EntireX Security secures distributed application components running with EntireX Broker. EntireX Security software is installed at specific points where communication between client and server / publish and subscribe application components is protected, using definitions located in the security repository of your organization: e.g., SAF-based security (RACF, CA ACF2 or CA Top Secret) under z/OS, and for UNIX and Windows either the local security system of the machine or an LDAP repository.

The basic functionality of EntireX Security covers

- authentication of user
- authorization of client and server, publish and subscribe, and Command and Information Services
- encryption of application

See *Functionality of EntireX Security*.

Overview of EntireX Security Features

Comprehensive Security

EntireX Security provides comprehensive security for EntireX Broker:

- user authentication
- user authorization

- application-data encryption
- supplied in object code only

Protection of Application Systems

EntireX Security protects client and server and publish and subscribe application systems, and, 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.

No User Exits to Write/Debug

There are no user exits to write and debug when using EntireX Security. Compare Sample Security Exits for Broker Security.

Standard Security Definitions

EntireX Security enables security definitions, based on class/name/service (client and server) or topic (publish and subscribe), to be validated within your SAF Security system. All definitions are managed using existing security procedures and software.

Protected Investment in SAF-based Security Repositories

Your investment in SAF-based security repositories is 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
- Authorization of Client and Server
- Authorization of Publish and Subscribe
- Authorization for Command and Information Services
- Encryption of Application Data

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. See *EntireX Security: Standard Security Solution*. It is the responsibility of the application to supply the ACI user ID and password on the first command. See `USER-ID` and `PASSWORD` under *Broker ACI Fields*.

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.

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 *EntireX Security: Standard Security Solution*) 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 *Administering Authorization Rules using System Management Hub* under UNIX | Windows.

Authorization of Publish and Subscribe

Authorization determines whether publisher and subscriber application components are allowed to execute with EntireX Broker. The topic name associated with the user's command forms the basis for the check. Separate authorization checks are performed, depending on the role of the application as either publisher or subscriber. The checks differentiate between a publisher's `PUBLISH` command and a subscriber's `SUBSCRIBE` 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 *EntireX Security: Standard Security Solution*.

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 *Administering Authorization Rules using System Management Hub* under UNIX | Windows.

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 shutdown a *service*, users must have the required authorization to register this class, server and service themselves.
- To shutdown 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.

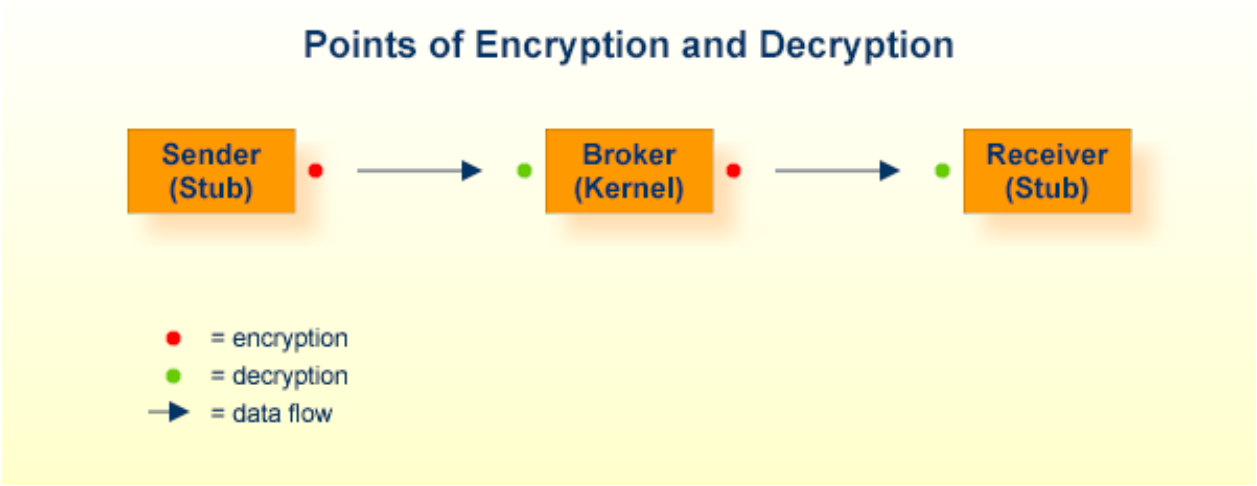
Similarly, Command and Information Services allows for third party subscription of users to a specified topic. In this case, a separate authorization check is made to ensure the issuing user is indeed authorized to subscribe to this same topic. This authorization is required in addition to the requesting 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 *Administering Authorization Rules using System Management Hub* under UNIX | Windows.

Encryption of Application Data

In EntireX Security, a client application can create and encrypt a message before sending it via a broker to a server application or vice versa. Similarly, publisher applications can encrypt messages before communicating them via a broker to subscriber applications. The following message flow illustrates an example client application sending a message to a server application. In a reverse message flow - that is, server application to client application - the points of encryption and decryption are also reversed.

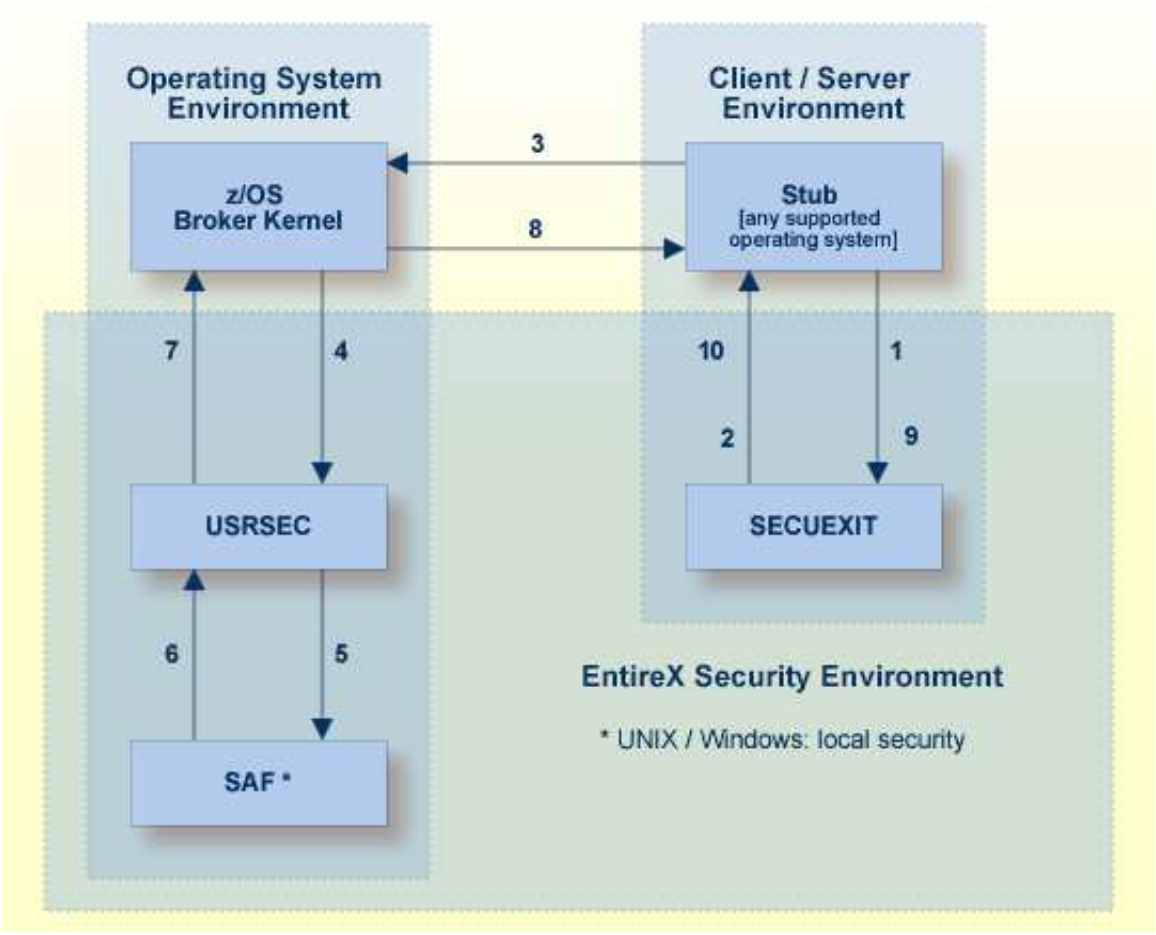
The setting of encryption by an application component activates encryption between that application and the broker kernel; it does not activate encryption between the broker kernel and the partner application component. However, the behavior of encryption from client via the broker to the server is controlled by the broker attribute `ENCRYPTION-LEVEL` and the control block field `ENCRYPTION-LEVEL`.



In the graphic, Sender refers to the message producer; Receiver refers to the message consumer.

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; authentication; encryption/decryption.



Note:
This diagram depicts the operation of the broker stub for Natural and 3GL. It is not intended to show the mechanism used by the Java ACI 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 and optionally the application data, based on the value assigned to the `ENCRYPTION-LEVEL` field of the broker control block, or through configuration options for: *z/OS* | UNIX | Windows.

3. Broker stub communicates the call to the broker kernel.
4. Broker kernel calls security module USRSEC, which provides the following functionality, based on the configuration options applicable to EntireX Security (see *Configuration Options for Broker*):

   - user authentication of publish and subscribe application components;
   - authorization checking required for subscriber to issue SUBSCRIBE command for specified topic;
   - authorization checking required for publisher to send publication;
   - encryption of application data;
   - decryption of application data;
   - 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

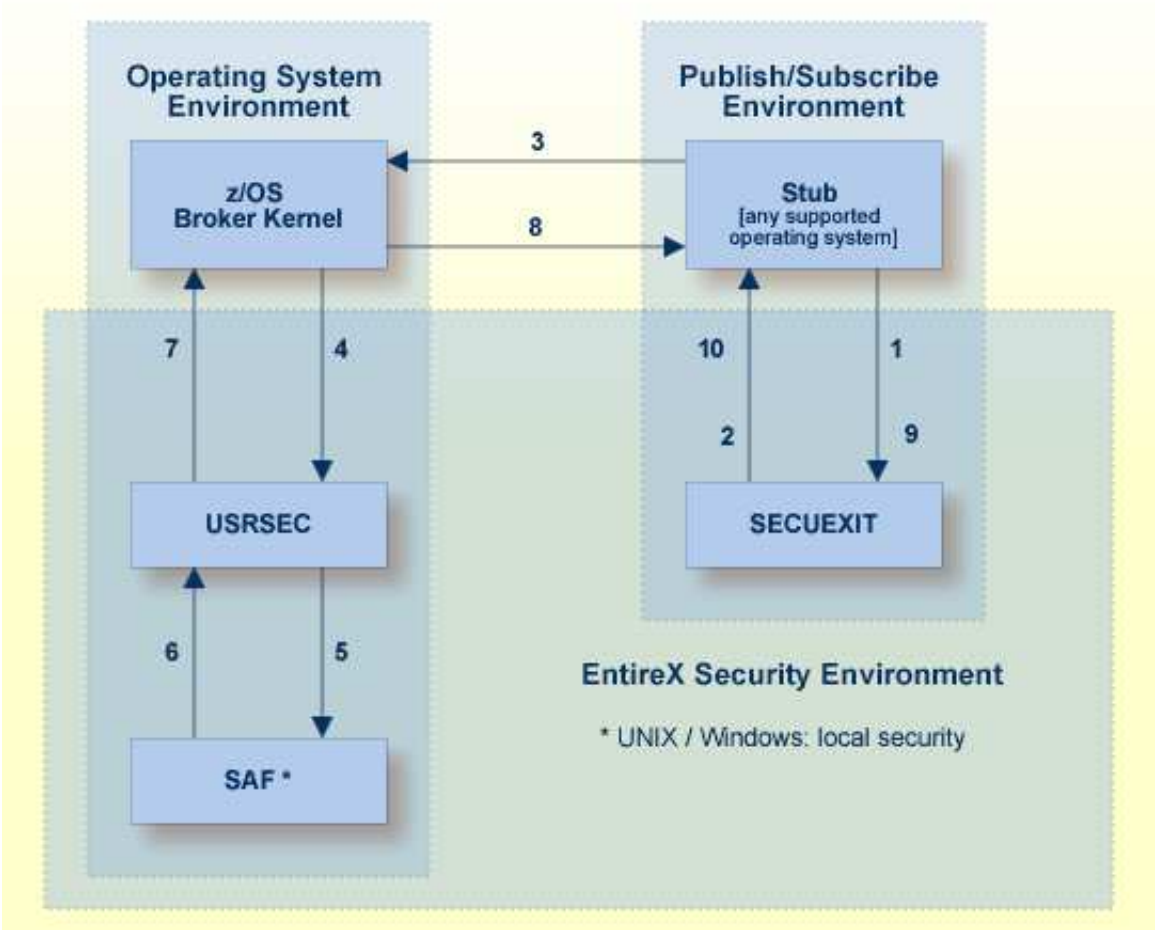
     Security module USRSEC calls SAF (RACF, CA ACF2 or CA Top Secret).
   - UNIX

     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. SECUEXIT performs decryption, where this is necessary to receive the data.
10. Security module SECUEXIT returns call to Broker stub.

Data Flow of EntireX Security (Publish and Subscribe)

The diagram shows the location of the security components of the kernel and stubs of Broker. Each step in the list below represents a specific step in the data flow sequence. The steps describe the functionality of the security components of the kernel / stubs of the broker: authorization; authentication; encryption/decryption.



Description of Steps in Data Flow

1. Broker stub calls security module `SECUEXIT`, if present.
2. Security module `SECUEXIT` encrypts the password and optionally the application data, based on the value assigned to the `ENCRYPTION-LEVEL` field of the broker control block, or through configuration options for: *z/OS* | UNIX | Windows.
3. Broker stub communicates the call to the broker kernel.
4. Broker kernel calls security module `USRSEC`, which provides the following functionality, based on the configuration options applicable to EntireX Security. See *Configuration Options for Broker*.
    - user authentication of client and server application components;

- authorization checking required for server to register a service;
- authorization checking required for client to send request;
- encryption of application data;
- decryption of application data;
- 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. SECUEXIT performs decryption, where this is necessary to receive the data.
10. Security module SECUEXIT returns call to Broker stub.

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;
- a publisher communicates a publication via broker for a specified topic;
- a subscriber issues the subscribe command to broker kernel for a specified topic;
- 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. *Encryption / Decryption* is performed in the kernel (as well as in the stub).

See *List of Components per Platform* 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 *List of Components per Platform* for where broker stubs are supported.

Encryption / Decryption

Encryption is the process by which the information or data being sent back and forth between two computers (including the password submitted when logging on) is "encoded", shielding it from view by unauthorized persons. With EntireX Security, the algorithms for encryption/decryption are present in the broker stubs and also in the kernel of broker.

See *Encryption of Application Data*.