webMethods EntireX

Broker

Version 10.9

April 2023

WEBMETHODS

This document applies to webMethods EntireX Version 10.9 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 © 1997-2023 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-BROKER-109-20230403

Table of Contents

1 About this Documentation ... 1
Document Conventions ... 2
Online Information and Support ... 2
Data Protection ... 3
I Concepts and Facilities of EntireX Broker ... 5
2 Concept of Interoperability ... 7
Interoperability and EntireX Broker ... 8
Messaging Model and Interoperability ... 8
Communication Models and Interoperability ... 11
3 General Architecture of EntireX Broker ... 13
Introduction to EntireX Broker Architecture ... 14
Client Server Communication Model ... 14
Architecture of Broker Stub ... 17
Architecture of Broker Kernel ... 19
4 Functionality of EntireX Broker ... 23
Application Bindings (Stubs) ... 24
Character Conversion ... 25
Command and Information Services ... 25
Accounting ... 26
Data Compression ... 26
Persistent Store ... 27
Units of Work ... 28
Security ... 29
5 Broker Quick Reference ... 31
ACI Syntax of Messaging Model ... 32
Location of Broker Kernel and Stubs ... 33
Transport: Broker Stubs and APIs ... 34
II Broker Attributes ... 37
6 Broker Attributes ... 39
Name and Location of Attribute File ... 41
Attribute Syntax ... 41
Broker-specific Attributes ... 43
Service-specific Attributes ... 62
Codepage-specific Attributes ... 73
Adabas SVC/Entire Net-Work-specific Attributes ... 76
Security-specific Attributes ... 79
TCP/IP-specific Attributes ... 86
c-tree-specific Attributes ... 89
SSL/TLS-specific Attributes ... 91
DIV-specific Attributes ... 96
Adabas-specific Attributes ... 98
Application Monitoring-specific Attributes ... 100
Authorization Rule-specific Attributes ... 101

Variable Definition File 102
III Broker Command and Information Services 103
7 Broker Command and Information Services 105
CIS Overview Table 106
Modes of Requesting the Services 107
ETBCMD: Executable Command Requests 109
ETBINFO: Returnable Information Requests 114
IV EntireX Broker Reporting 115
8 Command Logging in EntireX 121
Introduction to Command Logging 122
Command Log Filtering using Command-line Interface ETBCMD 125
ACI-driven Command Logging 127
Dual Command Log Files 127
V Building an EntireX Broker Image 129
9 Building an EntireX Broker Image 131
Prerequisites 132
Building and Running the EntireX Broker Image 132
Verifying the Build 136
Healthcheck for EntireX Broker 137

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

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.

I Concepts and Facilities of EntireX Broker

EntireX Broker is a middleware infrastructure that allows application components in a distributed processing environment to communicate with each other. EntireX Broker provides access through the *client and server* communication model. Message queues are employed to provide verifiable delivery of message data in asynchronous communication.

Additionally, EntireX Broker allows each application component to use a different programming interface. As a result, your application components can achieve highly flexible interoperability in a loosely coupled way. EntireX Broker can be used where your application components are located on distributed machines and where different operating systems and TP monitors are used on each machine.

Concept of Interoperability	Introduces the basic concept of EntireX Broker: achieving highly flexible interoperability of distributed application components.
General Architecture of EntireX Broker	Describes the components and transport mechanisms of EntireX Broker within the context of EntireX.
Functionality of EntireX Broker	Provides a brief overview of the functionality provided by EntireX Broker.
Broker Quick Reference	Quick Reference to Broker features and functions.

2 Concept of Interoperability

- Interoperability and EntireX Broker 8
- Messaging Model and Interoperability 8
- Communication Models and Interoperability 11

Interoperability and EntireX Broker

This section introduces the basic concept of EntireX Broker: achieving highly flexible interoperability of application components in a distributed processing environment. This concept is described from the perspectives of

- a messaging model
- communication models
- application programming interfaces
- EntireX components

in order to give you a comprehensive, high-level view of how EntireX Broker enables flexible interoperability between distributed application components.

Messaging Model and Interoperability

Introduction

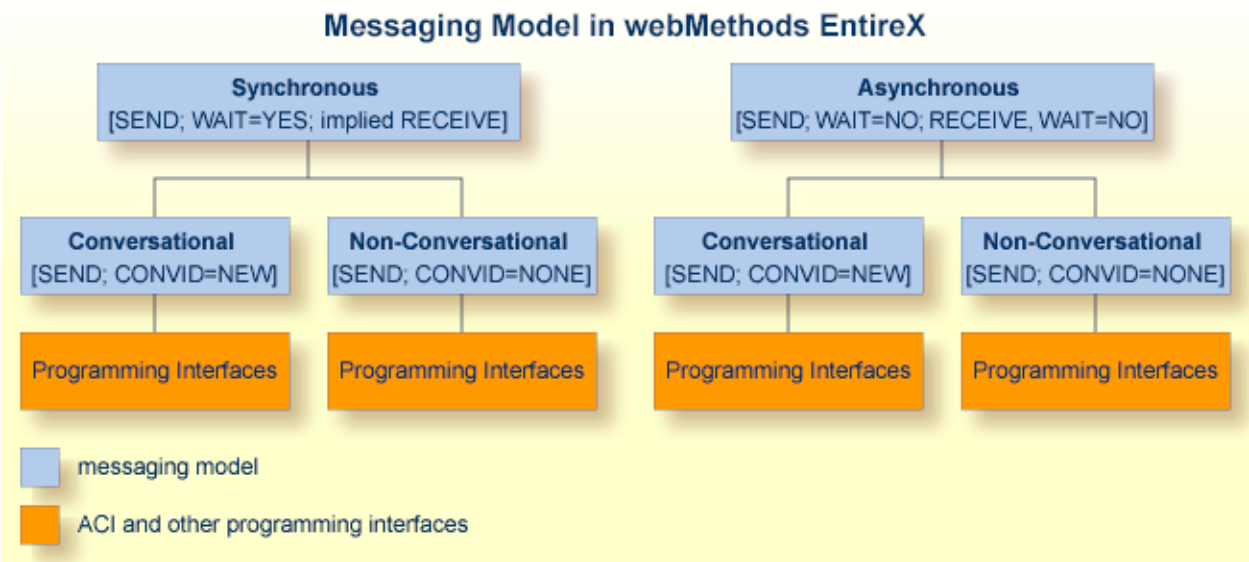
In a distributed processing environment that uses EntireX Broker, communication occurs through application components exchanging messages. An application component offering a service registers it with EntireX Broker (see `REGISTER`); this makes the service available to other application components able to communicate with EntireX Broker. An application component intending to access a service issues its request through EntireX Broker, which then routes the request to the specific application component offering the service.

The following concepts help describe how message exchange is structured in EntireX Broker:

- **Synchronicity**
  The application initiating the request either waits for the result to return, whereby it suspends all processing (synchronous); or it does not wait for the result to return, whereby it is freed to do other processing (asynchronous).
- **Conversationality**
  The request can either be a single pair of messages comprising request/reply (non-conversational); or it can be a sequence of multiple messages which are all part of the same request (conversational).

Overview Diagram

The following diagram shows the two major concepts of EntireX Broker's messaging model: synchronicity and conversationality. See *ACI Syntax of Messaging Model* below for a description of the messaging syntax.



ACI Syntax of Messaging Model

The table below describes the messaging terms mentioned in the diagram above from the viewpoint of the application component initiating the request, as expressed in ACI syntax.

The ACI (Advanced Communication Interface) is the lowest level application programming interface that interacts with EntireX Broker. The ACI is common to all of the messaging models and communication models (see ***Communication Models and Interoperability***) of EntireX.

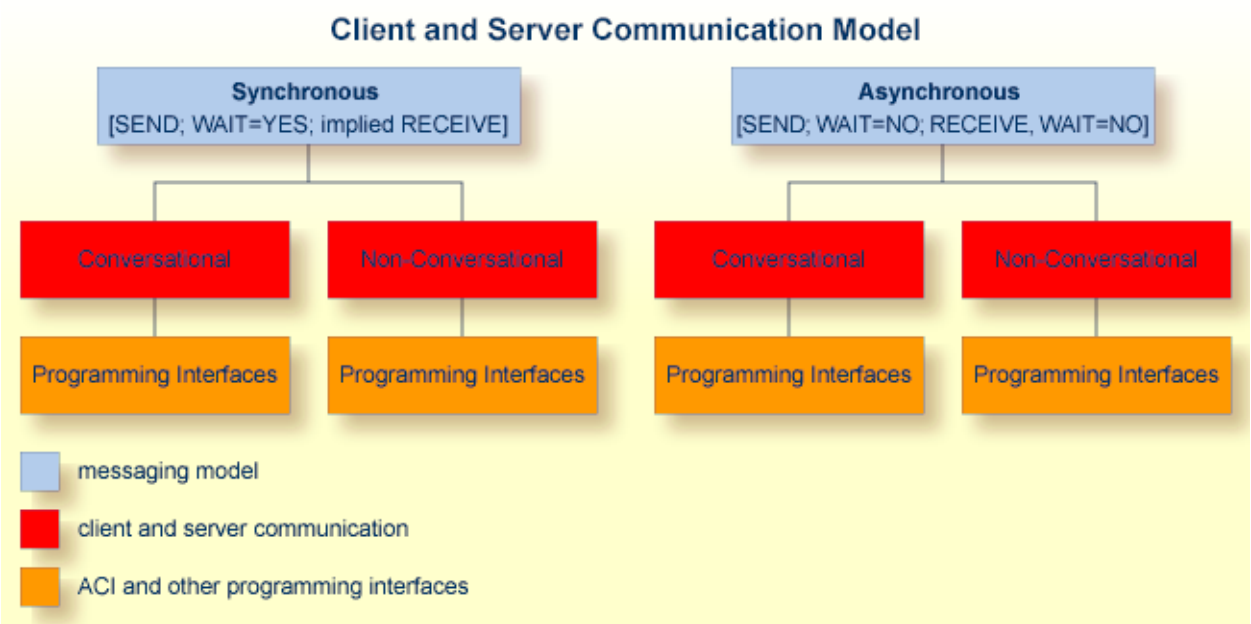
Messaging Term		Client	Server
Synchronicity	Synchronous	■ `SEND` [1] ■ `WAIT=YES` [1]	■ `RECEIVE` ■ `WAIT=YES`
	Asynchronous [2]	■ `SEND` ■ `WAIT=NO` ■ `WAIT=YES`	■ `RECEIVE` ■ `WAIT=NO`
Conversationality	Conversational [2]	■ `SEND` ■ `CONV-ID=NEW`	■ `RECEIVE`
	Non-conversational [2]	■ `SEND` ■ `CONV-ID=NONE`	■ `RECEIVE`

Notes:

1. The synchronous `SEND, WAIT=YES` command contains an implied `RECEIVE` command.
2. Persistence available. See *Concepts of Persistent Messaging* in the platform-independent Administration documentation.

Communication Models and Interoperability

The EntireX Broker uses the communication model client-and-server. This model is based on the connection between exactly two partners: client and server. This model covers the requirements of conversational communication and asynchronous processing.



3 General Architecture of EntireX Broker

- Introduction to EntireX Broker Architecture 14
- Client Server Communication Model 14
- Architecture of Broker Stub 17
- Architecture of Broker Kernel 19

Introduction to EntireX Broker Architecture

This section describes the command process flows within the Broker kernel and stubs when two application components communicate with each other using EntireX Broker. The Broker consists of the following components:

- a stub (application binding), which resides within the process space of each application component
- a Broker kernel, which resides in a separate process space, managing all the communication between application components

The details of the transport protocols remain transparent to the application components because they reside within EntireX Broker (stubs and kernel). The EntireX Broker kernel and the location of the transport protocols are the architectural aspects of EntireX Broker that distinguish it from other messaging middleware.

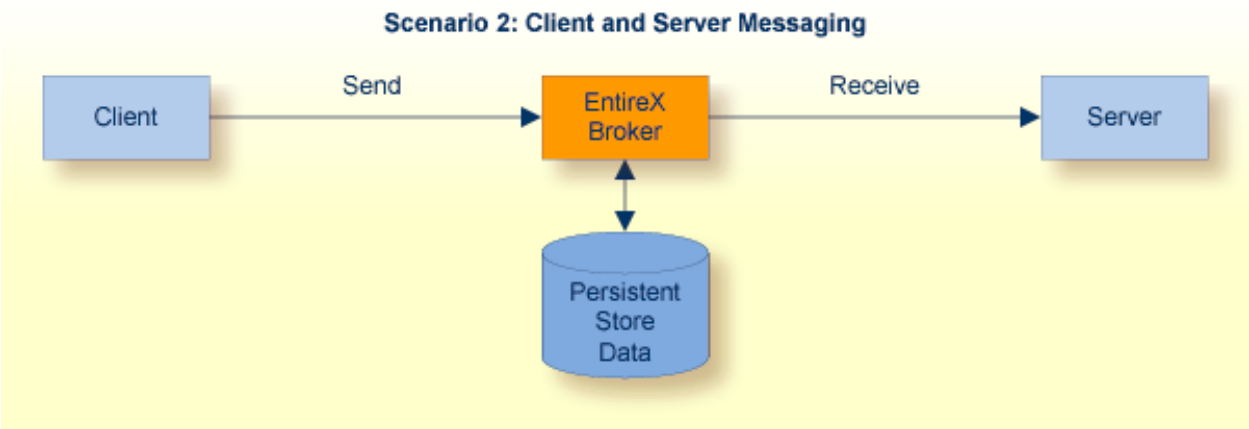
Client Server Communication Model

The EntireX Broker uses the communication model client and server. See *Writing Client and Server Applications* in the ACI Programming documentation for details.

Example Scenario 1: Client and Server Messaging (Synchronous)

This is a synchronous messaging scenario: send request and wait for a response.

Example Scenario 2: Client and Server Messaging (Asynchronous)



This is an asynchronous messaging scenario: put message in service queue.

Note that the terms “client” and “server” have specific meanings within the context of EntireX:

Client

An application component intending to access a service makes its request via EntireX Broker, which routes the request to the specific application component offering this service.

The request can be a single pair of messages comprising request/reply; or it can be a sequence of multiple, related messages containing one or more requests and one or more replies, known as a conversation. This enables EntireX Broker to be used for applications supporting different programming interfaces. It also allows interoperability between types of application components employing these different interfaces.

Server

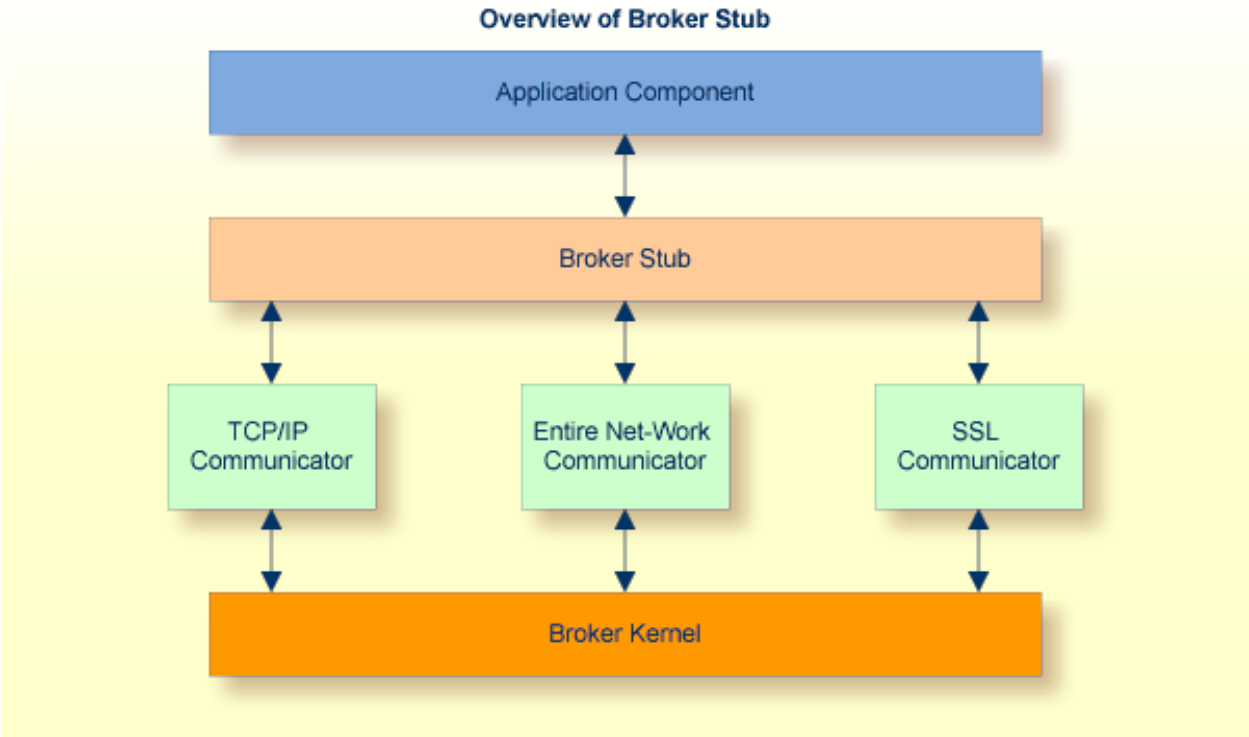
An application component offering a service registers it with EntireX Broker. EntireX Broker makes the registered service available to other application components capable of communicating with EntireX Broker. The fact that a server has been registered and is available in this way defines it as a service in terms of class/name/server within the context of EntireX.

Architecture of Broker Stub

The type of communication model described in this section and in the section *Architecture of Broker Kernel* is client and server.

Overview of Broker Stub

The EntireX Broker stub is another name for Software AG's ACI (Advanced Communication Interface). The stub implements an API (application programming interface) that allows programs written in various languages to access EntireX Broker.



See also *Administering Broker Stubs* in the platform-specific Administration documentation.

Description of Command Process Flow within Broker Stub

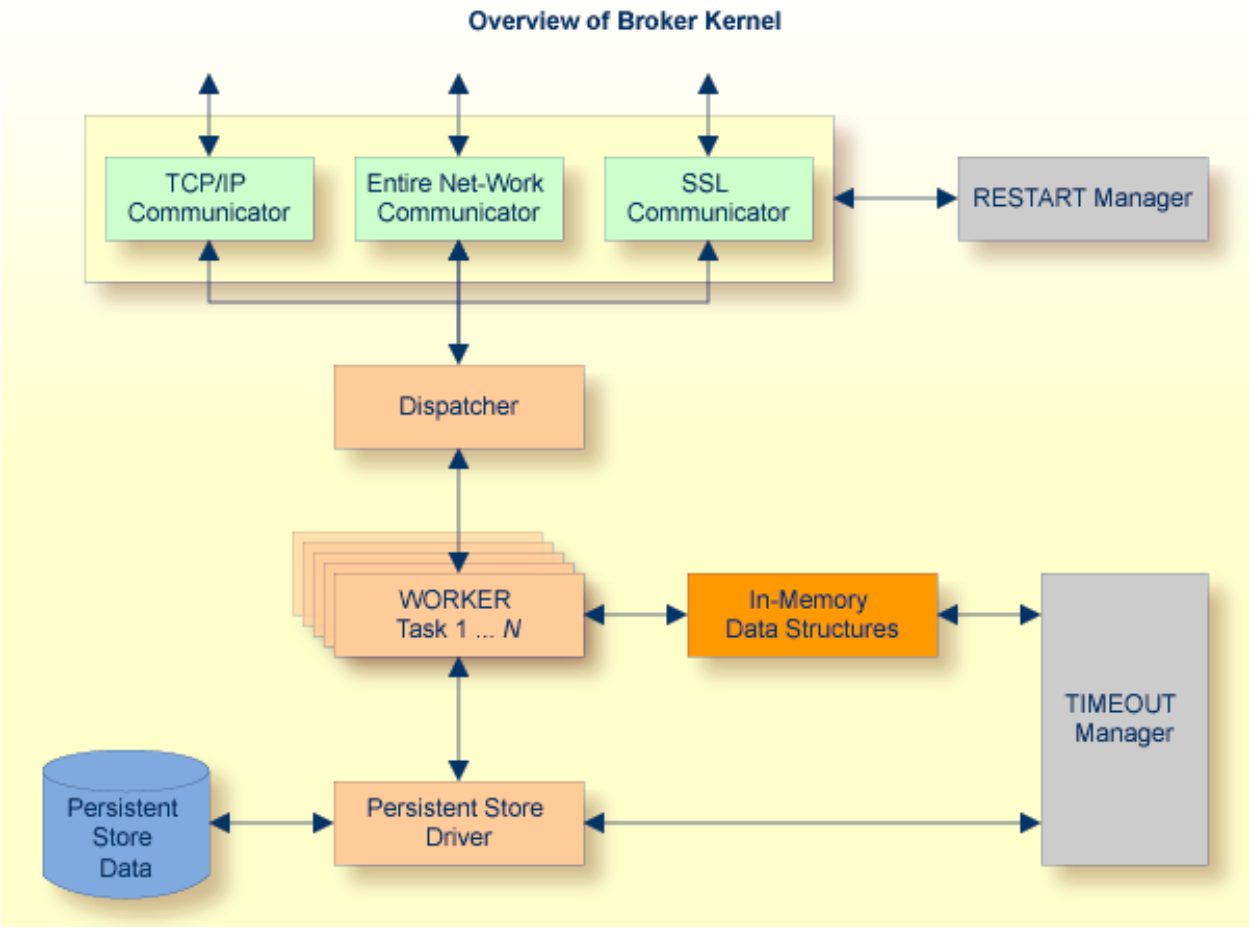
The following table gives a step-by-step description of a typical command process flow from and to a Broker stub. This example describes a `SEND/RECEIVE` command pair.

Step	Description
1	The originating application program calls the stub with a `SEND/WAIT=YES` command. The stub builds the necessary information structures and communicates the message to the Broker kernel. Basic validation is performed in the stub before the command is passed to the Broker kernel.
2	The stub uses one of the following transport mechanisms to transmit the command to the Broker kernel: TCP, SSL or Entire Net-Work. The application does not have to recognize the details of the transport protocol since all transport protocol processing resides entirely within the stub.
3	The application is suspended while the stub waits for a response. Since the application has issued `SEND`, `WAIT=YES` it must wait for the message to travel via the Broker kernel to the partner application which will satisfy the request.
4	After the request has been satisfied and the message returns from the partner application, via the Broker kernel, the stub will pass control back to the originating application.

Architecture of Broker Kernel

The type of communication model described in this section and in the section *Architecture of Broker Stub* is client and server.

Overview of Broker Kernel



Description of Command Process Flow within Broker Kernel

The following table gives a step-by-step description of a typical command process flow within the Broker kernel. This example describes a `SEND/RECEIVE` command pair.

Step	Description
1	The originating application program calls the Broker stub with a `SEND` command. The stub builds the necessary information structures and transmits the message to the Broker kernel using TCP, SSL or Entire Net-Work.
2	The message is received by one of the communications subtasks running within the Broker kernel. The communications subtask passes the message to the dispatcher.
3	The dispatcher schedules the processing of the message within a worker task inside the Broker kernel.
4	Worker task processes the inbound message, performing any necessary character conversion and security operations, and then determines the partner to which the message is to be routed. Any necessary persistence operations are performed under control of the worker task.
5	The outbound message is passed to the relevant communications subtasks within the Broker kernel for transmission to the partner application component.
6	The partner application component which has issued a `RECEIVE` command via the broker stub obtains the message from the originating application program.
7	The partner application component then processes the message and normally makes a reply.

Notes:

1. Application components can exchange successive related message pairs. This action constitutes a conversation.
2. Clean-up processing of timed-out commands is performed asynchronously by the Broker kernel Timeout Manager which acts upon in-memory data structures as well as data within the persistent store.
3. The communications restart manager is able to restart any communications subtasks which may have become temporarily disabled, for example by restarting the machine's TCP/IP driver.

4 Functionality of EntireX Broker

- Application Bindings (Stubs) 24
- Character Conversion 25
- Command and Information Services 25
- Accounting 26
- Data Compression 26
- Persistent Store 27
- Units of Work 28
- Security 29

This chapter gives an overview of the major value-added services provided by EntireX Broker. These services relieve the administrator or application builder of the task of providing the desired functionality.

Application Bindings (Stubs)

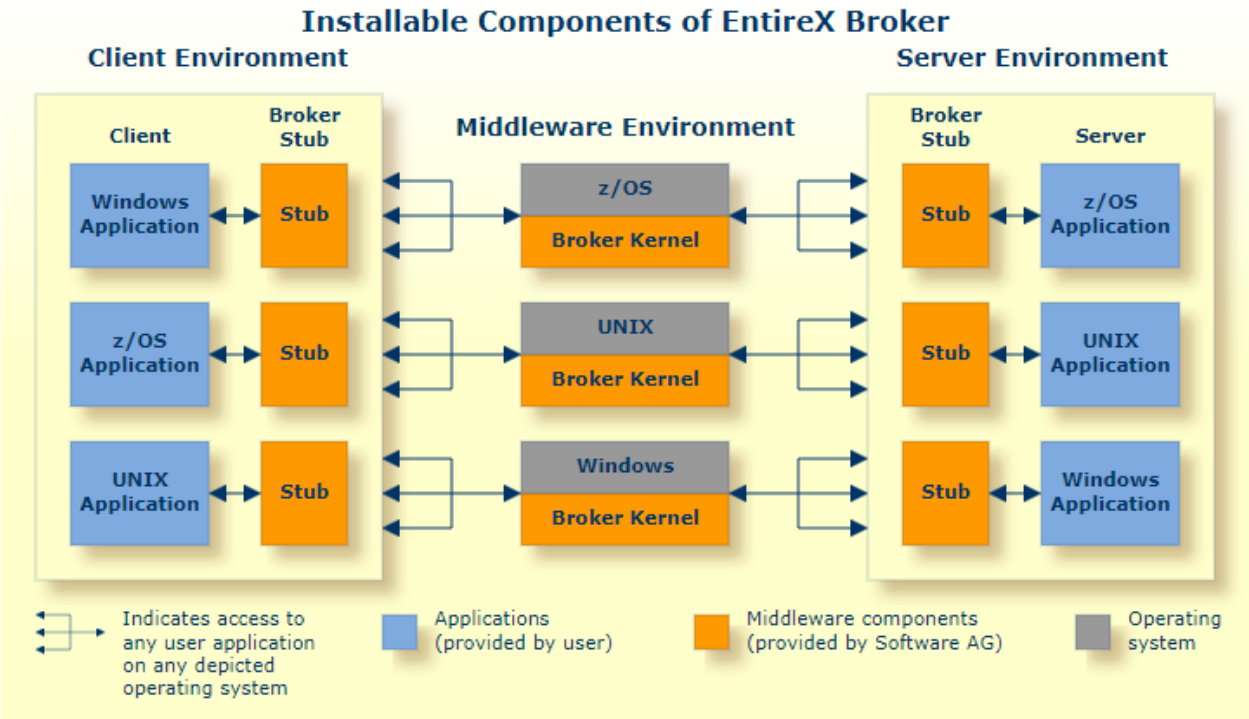
Application bindings allow applications developed in different programming languages and executing on various different platforms to be enabled by using EntireX Broker, see ***Architecture of Broker Stub***. Specifically, Java, Natural and other programs are easily enabled using EntireX Broker. These bindings are available on all major mainframe, Linux and Windows platforms.

The application binding is the glue between the application and the EntireX Broker kernel (see ***Architecture of Broker Kernel***, allowing your application to leverage all the functionality of EntireX regardless of

- programming language
- operating system
- hardware platform
- transport mechanism and
- choice of programming interfaces.

This binding capability enables various different application components to be integrated in a loosely coupled manner. See EntireX Java ACI and *EntireX Broker ACI* for Assembler | C | COBOL | Natural | PL/I | RPG.

Applications on z/OS, Linux, Windows etc. communicating with each other using stubs:



Character Conversion

Character conversion within the EntireX Broker means the incoming data is converted to the encoding of the target platform, using the codepages of the caller and receiver. See *Internationalization with EntireX*.

Command and Information Services

EntireX Broker includes a set of monitoring and control functions that enable you to monitor system resource utilization and view the current activities of the clients and servers on the system. These services are available through a Web-based interface, in addition to a command-line tool. An interface exists to allow program access to these facilities.

Accounting

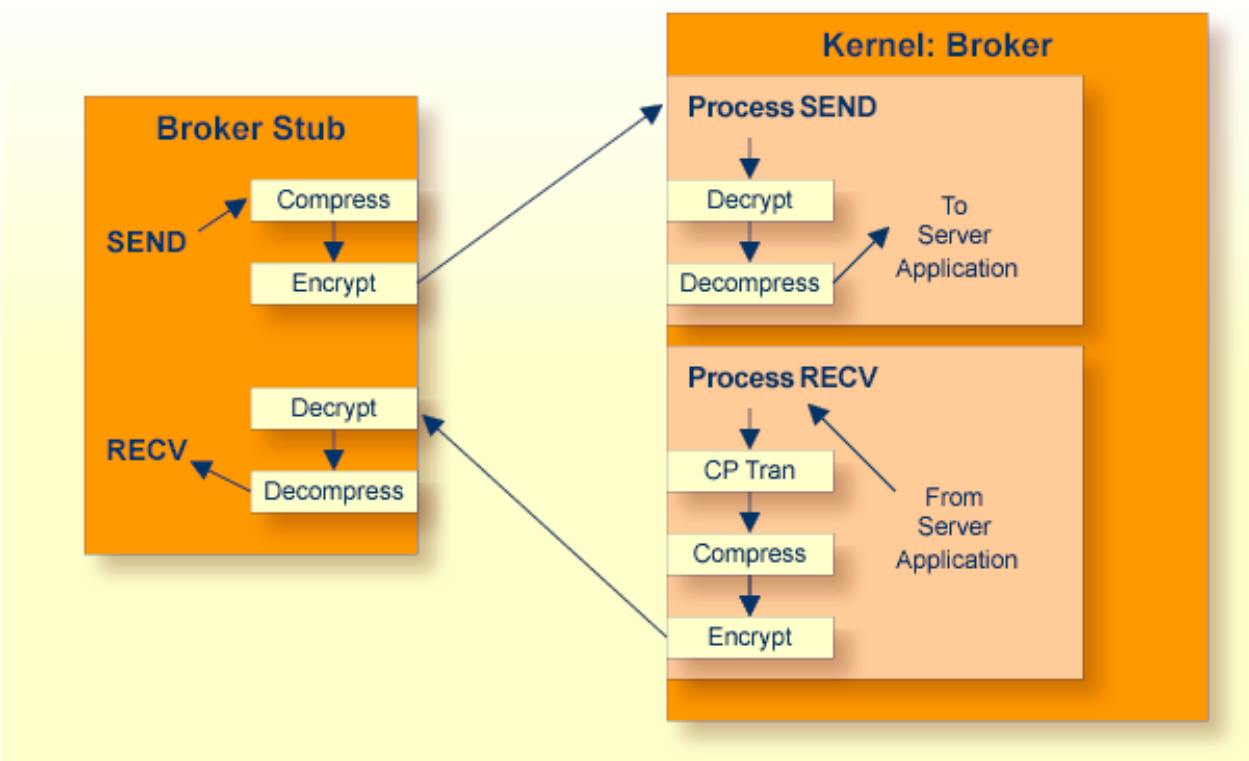
EntireX Broker provides accounting information based upon the flow of message sequences (or conversations). On z/OS, this information is written to standard accounting (SMF) records; on other platforms it is written to a file. The information can be used for:

- application chargeback: apportioning EntireX resource consumption on the conversation and/or the application level
- performance measurement: analyzing application throughput (bytes, messages, etc.) to determine overall performance
- trend analysis: using data to determine periods of heavy and/or light resource and/or application usage

Data Compression

EntireX allows compression of messages passed between application components so as to consume less network bandwidth. This is done independently of transport mechanism by compressing the message in the application binding before it is transmitted to the ***Broker Kernel***. The Broker kernel decompresses the message to enable security and data conversion to be applied.

The following graphic illustrates the sequencing of data compression within the stub and Broker kernel:



Persistent Store

The persistent store stores units of work for client and server applications.

Persistent message delivery ensures that messages sent between client and server (or server and client) application components can reach their target even in the event of application or system failures. The user application programs units of work to achieve persistent messaging. EntireX Broker provides persistent message delivery by grouping messages into units of work (UOWs) that are committed in one atomic operation by the sender. See also ***Units of Work***.

Persistence is implemented centrally within the ***Broker Kernel***. Therefore, the consistency of all the stored messages is guaranteed independently of the different application components and platforms from which the messages are derived.

Persistent Store Types

A persistent store driver is an executable, or a load module, which implements access to the physical persistent store. EntireX Broker allows the choice of three persistent store repositories: Adabas (DBMS), Data In Virtual (DIV) for z/OS, and native file system. The following table gives an overview of the persistent store options:

Persistent Store Type	Description	Operating System	Notes
Adabas	Uses Adabas database.	z/OS, Linux, Windows, BS2000	Adabas, Software AG's ADAptable dataBASe, is a high-performance, multithreaded, database management system.
DIV	Uses IBM Data In Virtual facility on z/OS.	z/OS	This persistent store option is implemented as a VSAM linear data set.
CTREE	c-tree© is an embedded local database that can be used as your persistent store.	Linux and Windows	c-tree© is the fast and reliable embedded database of FairCom Corporation®.

Units of Work

Units of work inform the sender of messages about their past and current status. Specifically, UOWs are used to:

- commit the sending of messages
- acknowledge the receipt of messages
- track the progress of sent messages at any point in time

Units of work are also the vehicle for achieving persistent messaging, although UOWs can be used without persistence.

See also *Using Units of Work* under *Using Persistence and Units of Work* in the platform-independent Administration documentation.

Security

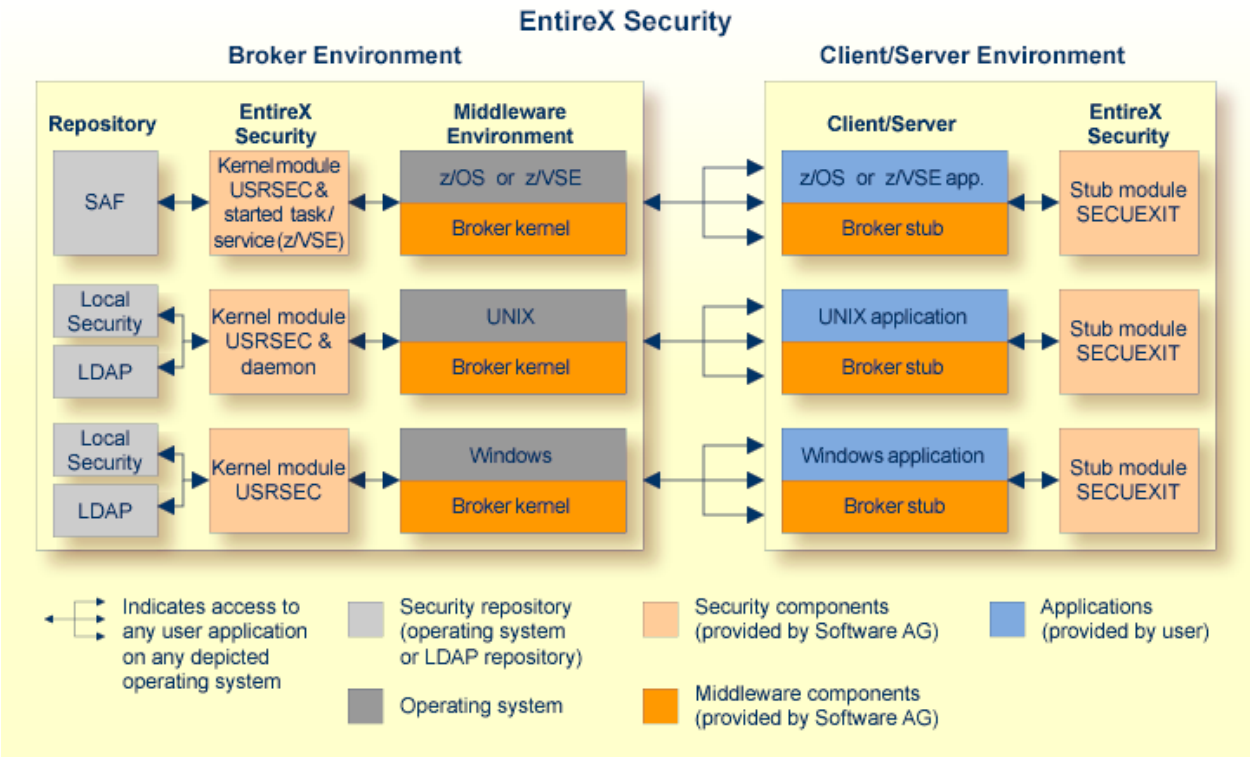
EntireX Security enables distributed application components running with Broker to be executed securely. EntireX Security is located centrally in the kernel of EntireX Broker giving it an overview of all messages sent between application components and therefore providing complete control over the authentication and authorization of each component.

Security checks are performed using a choice of security repositories, including:

- RACF
- CA ACF2
- CA Top Secret
- Linux and Windows security systems

The security repository chosen depends on the location of the Broker kernel. Because EntireX was designed to operate together with a security system, there is no additional application programming necessary.

This diagram depicts the location of the security components of the kernel and stubs of EntireX Broker:



See also *EntireX Security*.

5 Broker Quick Reference

- ACI Syntax of Messaging Model 32
- Location of Broker Kernel and Stubs 33
- Transport: Broker Stubs and APIs 34

ACI Syntax of Messaging Model

This table provides the ACI syntax used in EntireX Broker's communication model *Client and Server*.

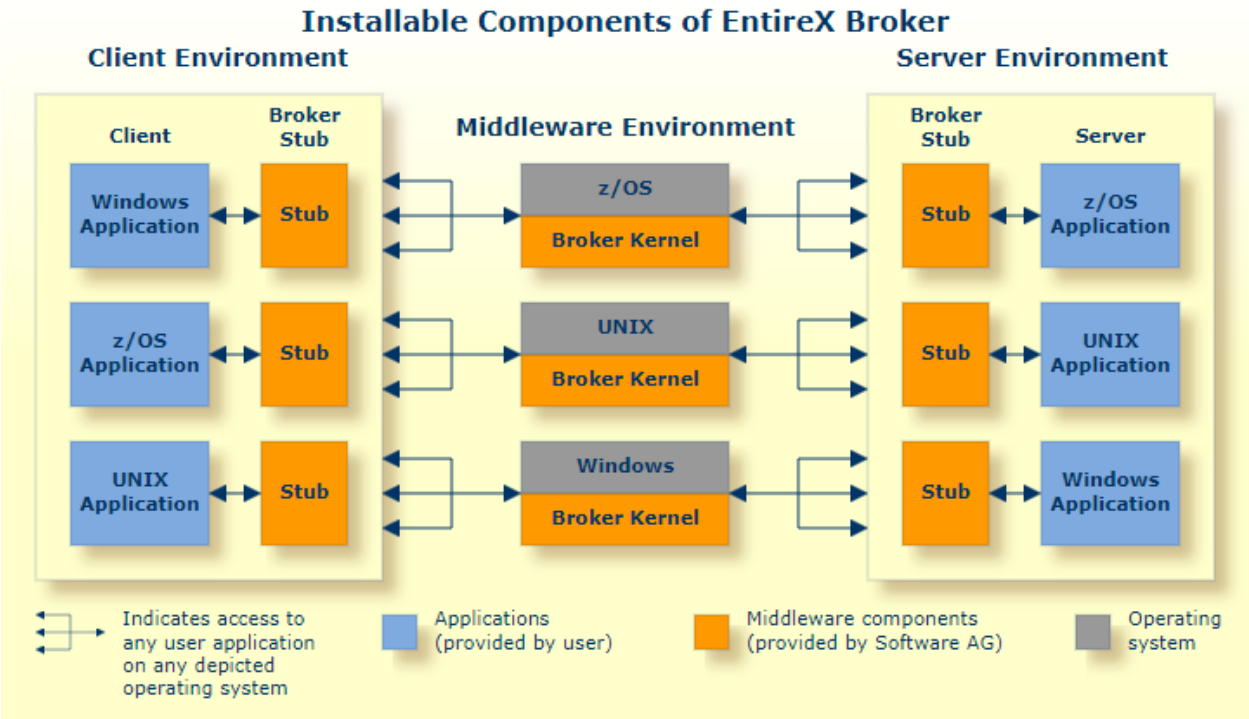
Messaging Term		Client	Server
Synchronicity	Synchronous	■ `SEND` (1) ■ `WAIT=YES` (1)	■ `RECEIVE` ■ `WAIT=YES`
	Asynchronous (2)	■ `SEND` ■ `WAIT=NO` ■ `WAIT=YES`	■ `RECEIVE` ■ `WAIT=NO`
Conversationality	Conversational (2)	■ `SEND` ■ `CONV-ID=NEW`	■ `RECEIVE`
	Non-conversational (2)	■ `SEND` ■ `CONV-ID=NONE`	■ `RECEIVE`

Notes:

1. The synchronous `SEND, WAIT=YES` command contains an implied `RECEIVE` command.
2. Persistence available. See *Concepts of Persistent Messaging* in the platform-independent Administration documentation.

Location of Broker Kernel and Stubs

This graphic shows the locations where the broker kernel and broker stubs can be installed. See *Architecture of Broker Kernel* and *Architecture of Broker Stub*.



Transport: Broker Stubs and APIs

This table gives an overview of the transport methods supported by EntireX Broker stubs.

Operating System	Environment	Module	Transport to Broker			
			TCP	SSL	NET [1]	HTTP(S) [5]
z/OS	Adabas Replication Services	ARFETB	x	[2]	x	
	Batch, TSO, IMS (BMP)	BROKER	x	[2]	x	
	Com-plete	COMETB	x	[2]	x	
	CICS	CICSETB	x	[2]	x	
	IMS (MPP)	MPPETB	x	[2]	x	
	IDMS/DC [3]	IDMSETB	x	[2]		
	Natural	NATETB23	x	[2]	x	
	Natural RPC Server	NATETBZ	x	[2]	x	
	z/OS UNIX System Services	EntireX Java ACI	x	x		x
Linux		broker.so	x	x		
		EntireX Java ACI	x	x		x
Windows		broker.dll [4]	x	x		
		EntireX Java ACI	x	x		x
BS2000	Batch, Dialog (formerly TIAM)	BROKER	x		x	
z/VSE	Batch	BKIMB	x	[6]	x	
	CICS	BKIMC	x	[6]	x	
IBM i		EXA	x			

Notes:

1. NET is available for transport to a broker running under mainframe platforms only; not to a broker running under Linux or Windows.
2. Under z/OS, use IBM's Application Transparent Transport Layer Security (AT-TLS). Refer to the IBM documentation for more information. See also *SSL/TLS, HTTP(S), and Certificates with EntireX*.
3. Tracing and transport timeout are not supported in this environment.
4. Stub broker32.dll is supported for reasons of backward compatibility. The functionality is identical to broker.dll.
5. Via EntireX Broker HTTP(S) Agent; see Broker HTTP(S) Agent in the platform-specific Administration documentation.

6. Under z/VSE, use BSI's Automatic Transport Layer Security (ATLS). Refer to the *BSI SSL Installation, Programming and User's Guide*. See also *SSL/TLS, HTTP(S), and Certificates with EntireX*.

See also:

- *Transport Methods for Broker Stubs* under z/OS | Linux | Windows | BS2000 | z/VSE in the platform-specific Administration documentation
- *Setting Transport Methods* under *Writing Advanced Applications - EntireX Java ACI*

II Broker Attributes

6 Broker Attributes

- Name and Location of Attribute File 41
- Attribute Syntax 41
- Broker-specific Attributes 43
- Service-specific Attributes 62
- Codepage-specific Attributes 73
- Adabas SVC/Entire Net-Work-specific Attributes 76
- Security-specific Attributes 79
- TCP/IP-specific Attributes 86
- c-tree-specific Attributes 89
- SSL/TLS-specific Attributes 91
- DIV-specific Attributes 96
- Adabas-specific Attributes 98
- Application Monitoring-specific Attributes 100
- Authorization Rule-specific Attributes 101
- Variable Definition File 102

The Broker attribute file contains a series of parameters (attributes) that control the availability and characteristics of clients and servers, as well as of the Broker itself. You can customize the Broker environment by modifying the attribute settings.

Note: This section lists all EntireX Broker parameters. Not all parameters are applicable to all supported operating systems.
For attributes applicable to a specific operation system, see the corresponding page in the z/OS | Linux | Windows | BS2000 Administration documentation.

Note: This section lists EntireX Broker parameters applicable to operating system z/OSLinux-WindowsBS2000 only. For attributes applicable to *all* operating systems, see *Broker Attributes* in the platform-independent Administration documentation.

Name and Location of Attribute File

The name and location of the broker attribute file is platform-dependent.

Platform	File Name/Location
z/OS	Member *EXBATTR* in the EntireX Broker source library.
Linux	File *etbfile* in directory <*InstDir*>/EntireX/config/etb/<*BrokerName*> (default). *
Windows	File <*BrokerName*>.atr in directory <*InstDir*>\EntireX\config\etb\<*BrokerName*> (default). *
BS2000	File *ETB-ATTR* in library *EXX103.JOBS*.

* When starting a broker manually, name and location of the broker attribute file can be overwritten with the environment variable `ETB_ATTR`.

Attribute Syntax

Each entry in the attribute file has the format:

```
ATTRIBUTE-NAME=value
```

The following rules and restrictions apply:

- A line can contain multiple entries separated by commas.
- Attribute names can be entered in mixed upper and lowercase.
- Spaces between attribute names, values and separators are ignored.
- Spaces in the attribute names are not allowed.
- Commas and equal signs are not allowed in value notations.
- Lines starting with an asterisk (*) are treated as comment lines. Within a line, characters following an * or # sign are also treated as comments.
- The `CLASS` keyword must be the first keyword in a service definition.
- Multiple services can be included in a single service definition section. The attribute settings will apply to all services defined in the section.
- Attributes specified after the service definition (`CLASS`, `SERVER`, `SERVICE` *keywords*) overwrite the default characteristics for the service.
- Attribute values can contain variables of the form `${variable name}` or `$variable name`:
  - Due to variations in EBCDIC codepages, braces should only be used on ASCII (Linux or Windows) platforms or EBCDIC platforms using the IBM-1047 (US) codepage.

- The variable name can contain only alphanumeric characters and the underscore (_) character.
- The first non-alphanumeric or underscore character terminates the variable name.
- Under Linux and Windows, the string `${variable name}` is replaced with the value of the corresponding environment variable.
- On z/OS, variable values are read from a file defined by the DD name `ETBVARS`. The syntax of this file is the same as the attribute file.
- If a variable has no value: if the variable name is enclosed in braces, error 00210594 is given, otherwise `$variable name` will be used as the variable value.
- If you encounter problems with braces (and this is quite possible in a z/OS environment), we suggest you omit the braces.

Broker-specific Attributes

The broker-specific attribute section begins with the keyword `DEFAULTS=BROKER`. It contains attributes that apply to the broker. At startup time, the attributes are read and duplicate or missing values are treated as errors. When an error occurs, the broker stops execution until the problem is corrected.

Tip: To avoid resource shortages for your applications, be sure to specify sufficiently large values for the broker attributes that define the global resources.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
ABEND-LOOP-DETECTION	YES | NO	O	z	u	w	b
	YES Stop broker if a task terminates abnormally twice, that is, the same abend reason at the same abend location already occurred. This attribute prevents an infinite abend loop. NO Use only if requested by Software AG Support. This setting may make sense if a known error leads to an abnormal termination, but a hotfix solving the problem has not yet been provided. Reset to YES when the hotfix has been installed.					
ABEND-MEMORY-DUMP	YES | NO	O	z	u	w	b
	YES Print all data pools of the broker if a task terminates abnormally. This dump is needed to analyze the abend. NO If the dump has already been sent to Software AG, you can set to NO to avoid the extra overhead.					
ACCOUNTING	NO | 128-255	O	z			
	NO | YES[SEPARATOR=*char*]	O		u	w	b
	Determines whether accounting records are created. NO Do not create accounting records. *nnn* The SMF record number to use when writing the accounting records. YES Create accounting data. *char*= separator character(s). Up to seven separator characters can be specified using the SEPARATOR suboption, for example: ACCOUNTING = (YES, SEPARATOR=;) If no separator character is specified, the comma character will be used. See also *Accounting in EntireX Broker* in the platform-specific Administration documentation.					
ACCOUNTING-VERSION	1 | 2 | 3 | 4 | 5	O	z	u	w	b

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Determines whether accounting records are created. 1 Collect accounting information. This value is supported for reasons of compatibility with EntireX Broker 7.2.1 and below. 2 Collect extended accounting information in addition to that available with option 1. 3 Create accounting records in layout of version 3. 4 Create accounting records in layout of version 4. 5 Create accounting records in layout of version 5. This parameter applies when ACCOUNTING is activated.					
ACI-CONVERSION	YES | NO	O	z	u	w	b
	Determines the handling of ACI request and response strings of USTATUS. YES Convert ACI request and response strings with ICU. See *ICU Conversion* in the Internationalization documentation. NO Translate ACI request and response with internal translation table without support of national characters. See *Translation User Exit* in the Internationalization documentation. **Note:** This attribute was undocumented in EntireX versions prior to 10.3 and had default value NO. This meant that a translation user exit was used instead; this is no longer recommended.					
APPLICATION-MONITORING or APPMON	YES | NO	O	z	u	w	b
	Enable application monitoring in EntireX Broker. YES Enable application monitoring. NO Disable application monitoring. See the separate Application Monitoring documentation.					
AUTOLOGON	YES | NO	O	z	u	w	b
	YES LOGON occurs automatically during the first SEND or REGISTER. NO The application has to issue a LOGON call.					
AUTOSTART	NO | YES	O		u	w	
	This attribute defines the autostart behavior of a broker. NO Broker is *not* started automatically with the next system start. YES Broker is restarted automatically with the next system start.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Note: Prior to EntireX version 10.5 this was handled by the Broker Administration Service.					
BLACKLIST-PENALTY-TIME	5M | *n* | *n*S | *n*M | *n*H	R	z	u	w	b
	Define the length of time a participant is placed on the PARTICIPANT-BLACKLIST to prevent a denial-of-service attack. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647). *n*M Non-activity time in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). See *Protecting a Broker against Denial-of-Service Attacks* in the platform-specific Administration documentation.					
BROKER-ID	A32	R	z	u	w	b
	Identifies the broker to which the attribute file applies. The broker ID must be unique per machine. **Note:** The numerical section of the BROKER-ID is no longer used to determine the DBID in the EntireX Broker kernel with Entire Net-Work transport (NET). To determine the DBID, use attribute NODE in the DEFAULTS=NET section of the attribute file.					
CLIENT-NONACT	15M | *n* | *n*S | *n*M | *n*H	R	z	u	w	b
	Define the non-activity time for clients. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647). *n*M Non-activity time in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). A client that does not issue a broker request within the specified time limit is treated as inactive and all resources for the client are freed.					
CMDLOG	NO | YES	O	z	u	w	b
	NO Command logging will not be available in the broker. YES Command logging features will be available in the broker.					
CMDLOG-FILE-SIZE	1024 | *n*	O	z	u	w	b
	Defines the maximum size of the file that the command log is written to, in kilobytes. The value must be 1024 or higher. The default value is 1024. When one command log file grows to this size, broker starts writing to the other file. For more details, see *Command Logging in EntireX*.					
CONTROL-INTERVAL	60S | *n* | *n*S | *n*M | *n*H	O	z	u	w	b

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Defines the time interval of time-driven broker-to-broker calls. 1. It controls the time between handshake attempts. 2. The standby broker will check the status of the standard broker after the elapsed CONTROL-INTERVAL time. *n* Same as *n*S. *n*S Interval in seconds (max. 2147483647). *n*M Interval in minutes (max. 35791394). *n*H Interval in hours (max. 596523). The minimum value is 16 seconds. We strongly recommend the default value (60 seconds), except for very slow machines.					
CONV-DEFAULT	UNLIM | *n*	O	z	u	w	b
	Default number of conversations that are allocated for every service. UNLIM The number of conversations is restricted only by the number of conversations globally available. Precludes the use of NUM-CONVERSATION. *n* Number of conversations. This value can be overridden by specifying a CONV-LIMIT for the service. A value of 0 (zero) is invalid.					
DEFERRED	NO | YES	O	z	u	w	b
	Disable or enable deferred processing of units of work. NO Units of work cannot be sent to the service until it is available. YES Units of work can be sent to a service that is not up and registered. They will be processed when the service becomes available.					
DYNAMIC-MEMORY-MANAGEMENT	YES | NO	O	z	u	w	b
	YES An initial portion of memory is allocated at broker startup based on defined NUM-* attributes or internal default values if no NUM-* attributes have been defined. More memory is allocated without broker restart if there is a need to use more storage. Unused memory is deallocated. The upper limit of memory consumption can be defined by the attribute MAX-MEMORY. See *Dynamic Memory Management* under *Broker Resource Allocation* in the platform-independent Administration documentation. NO All memory is allocated at broker startup based on the calculation from the defined NUM-* attributes. Size of memory cannot be changed. This was the known behavior of EntireX 7.3 and earlier.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	If you run your broker with attribute DYNAMIC-MEMORY-MANAGEMENT=YES, the following attributes are not needed: ■ CONV-DEFAULT ■ NUM-CONV[ERSATION] ■ HEAP-SIZE ■ NUM-LONG[-BUFFER] ■ LONG-BUFFER-DEFAULT ■ NUM-SERVER ■ SERVER-DEFAULT ■ NUM-SERVICE-EXTENSION ■ SHORT-BUFFER-DEFAULT ■ NUM-SERVICE ■ NUM-CLIENT ■ NUM-SHORT[-BUFFER] ■ NUM-CMDLOG-FILTER ■ NUM-UOW|MAX-UOWS|MUOW ■ NUM-COMBUF ■ NUM-WQE **Caution:** However, if one of these attributes is defined, it determines the allocation size of that particular broker resource.					
DYNAMIC-WORKER-MANAGEMENT	NO | YES	O	z	u	w	b
	NO All worker tasks are started at broker startup. The number of worker tasks is defined by NUM-WORKER. After this initial step, no further worker tasks can be started. This is default and simulates the behavior of EntireX version 8.0 and earlier. YES As above, the initial portion of worker tasks started at broker startup is determined by NUM-WORKER. However, if there is a need to handle an increased workload, additional worker tasks can be started at runtime without restarting broker. Conversely, if a worker task remains unused, it is stopped. The upper and lower limit of running worker tasks can be defined by the attributes WORKER-MIN and WORKER-MAX. If you run broker with DYNAMIC-WORKER-MANAGEMENT=YES, the following attributes are useful to optimize the overall processing: ■ WORKER-MAX ■ WORKER-MIN ■ WORKER-NONACT ■ WORKER-QUEUE-DEPTH ■ WORKER-START-DELAY The attribute NUM-WORKER defines the initial number of worker tasks started during initialization. See *Dynamic Worker Management*.					
ETBCOM	NO | YES	O	z	u	w	
	YES | NO	O				b
	Bundles the output of the various broker tasks in task ETBCOM.					
FORCE	NO | YES	O		u		

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	NO Go down with error if IPC resources still exist. YES Clean up the left-over IPC resources of a previous run. **Note:** 1. If broker is started twice, the second instance will kill the first by removing the IPC resources. 2. For z/OS and BS2000, see separate attribute FORCE under *DEFAULTS=NET*.					
HEAP-SIZE	1024 | *n*	O	z	u	w	b
	Defines the size of the internal heap in KB. Not required if you are using DYNAMIC-MEMORY-MANAGEMENT. If you are *not* using dynamic memory management, we strongly recommend specifying - as a minimum - the default value of 1024 KB.					
ICU-CONVERSION	YES | NO	O	z	u	w	b
	Disable or enable ICU conversion. YES ICU is loaded and available for conversion. It is a prerequisite for CONVERSION=SAGTCHA and CONVERSION=SAGTRPC. NO ICU is not loaded and not available for conversion. CONVERSION=SAGTCHA and CONVERSION=SAGTRPC cannot be used. If any of the broker service definitions uses the character conversion approach *ICU Conversion*, that is, CONVERSION=SAGTCHA or CONVERSION=SAGTRPC, ICU-CONVERSION must be set to YES. If you are using only a user exit (see *User Exits* under *Introduction* in the Internationalization documentation) or CONVERSION=NO as character conversion approach for all your broker service definitions, ICU-CONVERSION can be set to NO. ICU requires additional storage to run properly. If ICU conversion is not needed, setting ICU-CONVERSION to NO will help to avoid unnecessary storage consumption.					
ICU-DATA-DIRECTORY	Folder or directory name in quotes.	O	z	u	w	
	The location where the broker searches for ICU custom converters. See *Building and Installing ICU Custom Converters* in the platform-specific Administration documentation.					
ICU-SET-DATA-DIRECTORY	YES | NO	O	z	u	w	
	Disable or enable ICU custom converter usage. YES The broker tries to locate ICU custom converters with the mechanism defined by the platform. See *Building and Installing ICU Custom Converters* in the platform-specific Administration documentation. NO Use of ICU custom converters is not possible.					
IPV6	YES | NO	O	z	u	w	b

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	`YES` Establish SSL and TCP/IP transport in IPv6 and IPv4 networks according to the TCP/IP stack configuration. `NO` Establish SSL and TCP/IP transport in IPv4 network only. This attribute applies to EntireX version 9.0 and above.					
`LONG-BUFFER-DEFAULT`	`UNLIM` | *n*	O	z	u	w	b
	Number of long buffers to be allocated for each service. `UNLIM` The number of long message buffers is restricted only by the number of buffers globally available. Precludes the use of `NUM-LONG-BUFFER`. *n* Number of buffers. This value can be overridden by specifying a `LONG-BUFFER-LIMIT` for the service. A value of 0 (zero) is invalid.					
`MAX-MEMORY`	`0` | *n* | *n*K | *n*M | *n*G | `UNLIM`	O	z	u	w	b
	Defines the upper limit of memory allocated by broker if `DYNAMIC-MEMORY-MANAGEMENT=YES` has been defined. `0`, `UNLIM` No memory limit. `others` Defines the maximum limit of allocated memory. If limit is exceeded, error 671 "Requested allocation exceeds MAX-MEMORY" is generated.					
`MAX-MESSAGE-LENGTH`	`2147483647` | *n*	O	z	u	w	b
	Maximum message size that the broker kernel can process. This value is transport-dependent. The default value represents the highest positive number that can be stored in a four-byte integer.					
`MAX-MESSAGES-IN-UOW`	`16` | *n*	O	z	u	w	b
	Maximum number of messages in a unit of work.					
`MAX-MSG`	See `MAX-MESSAGE-LENGTH`.					
`MAX-TRACE-FILES`	`4` | *n*	O		u	w	
	Defines the number of backup copies of the trace file ETB.LOG. Minimum number is 1; maximum is 999. A new trace file is allocated when the value for `TRACE-FILE-SIZE` is exceeded. These two attributes prevent a constantly growing ETB.LOG file. See *Trace File Handling* under Linux | Windows.					
`MAX-UOW-MESSAGE-LENGTH`	See `MAX-MESSAGE-LENGTH`.					
`MAX-UOWS`	`0` | *n*	O	z	u	w	b
	The maximum number of UOWs that can be concurrently active broker-wide. The default value is 0 (zero), which means that the broker will process only messages that are not part of a unit of work. If UOW processing is to be done by any service, a `MAX-UOWS` value must be 1 or larger for the broker.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	The MAX-UOWS value for the service will default to the value set for the broker. NUM-UOW is an alias of this parameter.					
MESSAGE-CASE	NONE | UPPER | LOWER	O	z	u	w	b
	Indicates if certain error message texts returned by the broker to its clients or written by the broker to its log file are to be in mixed case, uppercase, or lowercase. NONE No changes are made to message case. UPPER Messages are changed to uppercase. LOWER Messages are changed to lowercase.					
MUOW	See NUM-UOW.					
NEW-UOW-MESSAGES	YES | NO	O	z	u	w	b
	YES New UOW messages are allowed. NO New UOW messages are not allowed. This applies to UOW when using Persistence and should not be used for non-persistent UOWs. A usage example could be the following: The broker persistent store reaches capacity and the broker shuts down. You can set NEW-UOW-MESSAGES to NO to prevent new UOW messages from being added after a broker restart. This action allows only consumption (not production) of UOWs to occur after broker restart. After the persistent store capacity has been sufficiently reduced, the EntireX Broker administrator can issue a CIS command, see ALLOW-NEWUOWMSGS. This action allows new UOW messages to be sent to the broker. Reset attribute NEW-UOW-MESSAGES to YES, which permits new UOW messages to be produced in subsequent broker sessions.					
NUM-BLACKLIST-ENTRIES	256 | *n*	O	z	u	w	b
	Number of entries in the participant blacklist. Default value is 256 entries. Together with BLACKLIST-PENALTY-TIME and PARTICIPANT-BLACKLIST, this attribute is used to protect a broker running with SECURITY=YES against denial-of-service attacks. See *Protecting a Broker against Denial-of-Service Attacks* in the platform-specific Administration documentation.					
NUM-CLIENT	*n*	R	z	u	w	b
	Number of clients that can access the broker concurrently. A value of 0 (zero) is invalid.					
NUM-CMDLOG-FILTER	1 | *n*	O	z	u	w	b
	Maximum number of filters that can be specified simultaneously. **Tip:** We recommend you limit this value to the number of services that are being monitored. Minimum value is 1. A value of zero is invalid when the attribute CMDLOG is set to YES. See ***Command Logging in EntireX*** in the EntireX Broker documentation for more information.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
NUM-COMBUF	1024 | 1-999999	R	z	u	w	b
	Determines the maximum number of communication buffers available for processing commands arriving in the broker kernel. The size of one communication buffer is usually 16 KB split into 32 slots of 512 bytes, but it ultimately depends on the hardware architecture of your CPU. A value of 0 (zero) is invalid.					
NUM-CONVERSATION or NUM-CONV	*n* | AUTO	R	z	u	w	b
	Defines the number of conversations that can be active concurrently. The number specified should be high enough to account for both conversational and non-conversational requests. (Non-conversational requests are treated internally as one-conversation requests.) *n* Number of conversations. AUTO Uses the CONV-DEFAULT and the service-specific CONV-LIMIT values to calculate the number of conversations. Do not set the values used in the calculation to UNLIM. **Note:** 1. A value of 0 (zero) is invalid. If a wildcard service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 2. See ***Wildcard Service Definitions***.					
NUM-LONG-BUFFER or NUM-LONG	4096 | *n* | AUTO	R	z	u	w	b
	Defines the number of long message containers. Long message containers have a fixed length of 4096 bytes and are used to store requests that are larger than 2048 bytes. Storing a request of 8192 bytes, for example, would require two long message containers. *n* Number of buffers. AUTO Uses the LONG-BUFFER-DEFAULT and the service-specific LONG-BUFFER-LIMIT values to calculate the number of long message buffers. Do not set the values used in the calculation to UNLIM. A value of 0 (zero) is invalid. In *non-conversational* mode, message containers are released as soon as the client receives a reply from the server. If no reply is requested, message containers are released as soon as the server receives the client request. In *conversational* mode, the last message received is always kept until a new one is received. **Note:**					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	1. If a catch-all service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 2. See *Wildcard Service Definitions*.					
NUM-PARTICIPANT-EXTENSION	*n*	O	z	u	w	b
	Defines the number of participant extensions to link participants as clients and servers. *n* Number of participant extensions. *not specified* If this attribute is not set, the default value is calculated based on NUM-CLIENT and NUM-SERVER. A value of 0 (zero) is invalid.					
NUM-SERVER	*n* | AUTO	R	z	u	w	b
	Defines the number of servers that can offer services concurrently using the broker. This is *not* the number of services that can be registered to the broker (see NUM-SERVICE). *n* Number of servers. AUTO Uses the SERVER-DEFAULT and the service-specific SERVER-LIMIT values to calculate the number of servers. Do not set the values used in the calculation to UNLIM. **Note:** 1. Setting this value higher than the number of services allows the starting of server replicas that provide the same service. 2. A value of 0 (zero) is invalid. If a wildcard service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 3. See *Wildcard Service Definitions*.					
NUM-SERVICE	*n*	R	z	u	w	b
	Defines the number of services that can be registered to the broker. This is *not* the number of servers that can offer the services (see NUM-SERVER). A value of 0 (zero) is invalid.					
NUM-SERVICE-EXTENSION	*n* | AUTO	O	z	u	w	b
	Defines the number of service extensions to link servers to services. *n* Number of service extensions. AUTO Uses the value specified or calculated for NUM-SERVER + NUM-CLIENT, plus an extra cushion. *not specified* If this attribute is not set, the default value is NUM-SERVER multiplied by NUM-SERVICE.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	The minimum value is NUM-SERVER. The maximum value is NUM-SERVER multiplied by NUM-SERVICE. Caution is recommended with this attribute: ■ Set this attribute only if the storage resources allocated for service extensions need to be restricted. ■ Note that the value *n* allows only the specified number of server instances of *n* to be used. ■ Value AUTO will calculate the number of allowed server instances from NUM-SERVER, which itself might be set to AUTO. In this case, this also considers the value of SERVER-DEFAULT and even the individual SERVER-LIMIT for each service definition.					
NUM-SHORT-BUFFER or NUM-SHORT	*n* | AUTO	R	z	u	w	b
	Defines the number of short message containers. Short message containers have a fixed length of 256 bytes and are used to store requests of no more than 2048 bytes. To store a request of 1024 bytes, for example, would require four short message containers. *n* Number of buffers. AUTO Uses the SHORT-BUFFER-DEFAULT and the service-specific SHORT-BUFFER-LIMIT values to calculate the number of short message buffers. Do not set the values used in the calculation to UNLIM. **Note:** 1. In *non-conversational* mode, message containers are released as soon as the client receives a reply from the server. If no reply is requested, message containers are released as soon as the server receives the client request. 2. In *conversational* mode, the last message received is always kept until a new one is received. 3. If a wildcard service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 4. See ***Wildcard Service Definitions***.					
NUM-UOW	0 | *n*	O	z	u	w	b
	The maximum number of UOWs that can be concurrently active broker-wide. The default value is 0 (zero), which means that the broker will process only messages that are not part of a unit of work. If UOW processing is to be done by any service, a NUM-UOW value must be 1 or larger for the broker. (MAX-UOWS is an alias for this attribute.) The NUM-UOW value for the service will default to the value set for the broker.					
NUM-WORKER	1 | *n* (max. 64)	R	z	u	w	b

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Number of worker tasks that the broker can use. The number of worker tasks determines the number of functions (`SEND`, `RECEIVE`, `REGISTER`, etc.) that can be processed concurrently. At least one worker task is required; this is the default value.					
`NUM-WQE`	`1-32768`	R	z	u	w	b
	Maximum number of requests that can be processed by the broker in parallel, over all transport mechanisms. Each broker command is assigned a worker queue element, regardless of the transport mechanism being used. This element is released when the user has received the results of the command, including the case where the command has timed out.					
`PARTICIPANT-BLACKLIST`	`YES` | `NO`	R	z	u	w	b
	Determines whether participants attempting a denial-of-service attack on the broker are to be put on a blacklist. `YES` Create a participant blacklist. `NO` Do not create a participant blacklist. See *Protecting a Broker against Denial-of-Service Attacks* in the platform-specific Administration documentation.					
`PERCENTAGE-FOR-CONNECTION-SHORTAGE-MESSAGE`	`90` | `1-100`	O	z	u	w	b
	Broker will issue a message if the defined percentage value of TCP/IP connections (available file descriptors) is exceeded. Default is 90 percent of the available file descriptors.					
`POLL`	`YES` | `NO`	O	z	u		
	In earlier EntireX versions, the maximum number of TCP/IP connections per communicator was limited; see *Maximum TCP/IP Connections per Communicator* under *Broker Resource Allocation* for platform-specific list. With attribute `POLL` introduced in EntireX version 9.0, this restriction can be lifted under z/OS and Linux. `NO` This setting is used to run the compatibility mode in Broker. The `poll()` system call is not used. The limitations described under *Maximum TCP/IP Connections per Communicator* under *Broker Resource Allocation* apply. `YES` The `poll()` system call is used to lift the resource restrictions with `select()` in multiplexing file descriptor sets. **Note:** The maximum number of file descriptors per process is a hard limit that cannot be exceeded by `POLL=YES`. Setting this attribute to `YES` increases CPU consumption. `POLL=YES` is only useful if ■ you need more than the maximum number of TCP/IP connections per communicator, as described under *Maximum TCP/IP Connections per Communicator* under *Broker Resource Allocation*, and					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	■ this maximum number is less than the maximum number of file descriptors per process We recommend POLL=NO to reduce CPU consumption.					
POSTPONED-QUEUE	YES | NO	O	z	u	w	
	Enable or disable the creation of a postponed queue for Broker. YES Enable creation of a postponed queue. Define your postponed queue with service-specific attributes POSTPONE-ATTEMPTS and POSTPONE-DELAY. NO Disable creation of a postponed queue. See *Postponing Units of Work*.					
PSTORE	NO | HOT | COLD	O	z	u	w	b
	Defines the status of the persistent store at broker startup, including the condition of persistent units of work (UOWs). With any value other than NO, PSTORE-TYPE must be set. NO No persistent store. HOT Persistent UOWs are restored to their prior state during initialization. COLD Persistent UOWs are not restored during initialization, and the persistent store is considered empty. **Note:** For a hot or cold start, the persistent store must be available when your broker is restarted.					
PSTORE-REPORT	NO | YES	O	z	u	w	b
	Determines whether PSTORE report is created. NO Do not create the PSTORE report file. YES Create the PSTORE report file. See also *Persistent Store Report*.					
PSTORE-TYPE	DIV (z/OS) | CTREE (Linux, Windows) | ADABAS (all platforms)	O	z	u	w	b
	Describes the type of persistent store driver required. DIV Data in Virtual. z/OS only, and default on this platform. See ***DIV-specific Attributes*** below and *Implementing a DIV Persistent Store*. CTREE c-tree database. Linux and Windows only. See ***c-tree-specific Attributes*** and *c-tree Database as Persistent Store* in the Linux | Windows Administration documentation.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	ADABAS Adabas. All platforms. See also ***Adabas-specific Attributes*** (below) and *Managing the Broker Persistent Store* in the platform-specific Administration documentation.					
PSTORE-VERSION	5	O	z	u	w	b
	Determines the version of the persistent store. PSTORE-VERSION=5 is the only supported version since EntireX version 10.8. **Note:** To change the value of PSTORE-VERSION, the persistent store must be empty (all units of work must be consumed). If the persistent store is not empty, the start of the Broker with a changed PSTORE-VERSION may fail with error ETBE0741 or ETBM0745.					
SECURITY	NO | YES	O	z	u	w	b
	Determines whether EntireX Security is activated. NO EntireX Security is not activated. YES EntireX Security is activated. See *EntireX Security*.					
SERVER-DEFAULT	*n* | UNLIM	O	z	u	w	b
	Default number of servers that are allowed for every service. *n* Number of servers. UNLIM The number of servers is restricted only by the number of servers globally available. Precludes the use of NUM-SERVER=AUTO. This value can be overridden by specifying a SERVER-LIMIT for the service. A value of 0 (zero) is invalid.					
SERVICE-UPDATES	YES | NO	O	z	u	w	b
	Switch on/off the automatic update mode of the broker. YES The broker reads the attribute file whenever a service registers for the first time. This allows the broker to honor modifications in the attribute file *without* a restart. The attribute file is read only when the first server registers for a particular service; it is not reread when a second replica is activated. NO The attribute file is read only once during broker startup. Any changes to the attribute file will be honored only if the broker is restarted.					
SHORT-BUFFER-DEFAULT	UNLIM | *n*	O	z	u	w	b
	Number of short buffers to be allocated for each service.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	`UNLIM` The number of short message buffers is restricted only by the number of buffers globally available. Precludes the use of `NUM-SHORT-BUFFER=AUTO`. *n* Number of buffers. This value can be overridden by specifying a `SHORT-BUFFER-LIMIT` for the service. A value of 0 (zero) is invalid.					
`STORAGE-REPORT`	`NO` | `YES`	O	z	u	w	b
	Create a storage report about broker memory usage. `NO` Do not create the storage report. `YES` Create the storage report. See *Storage Report*.					
`STORE`	`OFF` | `BROKER`	O	z	u	w	b
	Sets the default `STORE` attribute for all units of work. This attribute can be overridden by the `STORE` field in the Broker ACI control block. `OFF` Units of work are not persistent. `BROKER` Units of work are persistent.					
`TRACE-DD`	`A255`	O	z			
	A string containing data set attributes enclosed in quotation marks. These attributes describe the trace output file and must be defined if you are using using a GDG (generation data group) as output data set. See *Flushing Trace Data to a GDG Data Set* under *Tracing EntireX Broker*. The following keywords are supported as part of the `TRACE-DD` value: ■ `DATACLAS` ■ `DCB` including `BLKSIZE`, `DSORG`, `LRECL`, `RECFM` ■ `DISP` ■ `DSN` ■ `MGMTCLAS` ■ `SPACE` ■ `STORCLAS` ■ `UNIT` Refer to your JCL Reference Manual for a complete description of the syntax. Example:					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000

	`TRACE-DD = "DSNAME=EXX.GDG,`
	`DCB=(BLKSIZE=1210,DSORG=PS,LRECL=121,RECFM=FB),`
	`DISP=(NEW,CATLG,CATLG),`
	`SPACE=(CYL,(100,10)),`
	`STORCLAS=SMS"`
	
	Note: If you specify `TRACE-DD`, you must also specify `TRMODE=WRAP` and a value for `TRBUFNUM` for the setting to take effect.					

`TRACE-FILE-SIZE`	*n* | *n*K | *n*M | *n*G	O		u	w	
	Defines the size of one trace file in kilobytes, megabytes or gigabytes. If this size is exceeded, a new trace file is allocated until the maximum number of trace files specified with `MAX-TRACE-FILES` is reached. There is no default value. These two parameters help prevent a constantly growing ETB.LOG file. See *Trace File Handling* under Linux | Windows.					
`TRACE-LEVEL`	`0-4`	O	z	u	w	b
	The level of tracing to be performed while the broker is running. `0` No tracing. Default value. `1` Traces incoming requests, outgoing replies, resource usage and conversion errors. `2` All of trace level 1, plus all main routines executed. `3` All of trace level 2, plus all routines executed. `4` All of trace level 3, plus Broker ACI control block displays. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the `TRACE-LEVEL` attribute, you must restart the broker for the change to take effect. For temporary changes to `TRACE-LEVEL` without a broker restart, use Command Central or the EntireX Broker command-line utility `ETBCMD`.					
`TRANSPORT`	`TCP-NET` | `TCP` | `SSL` | `NET`	O	z			b
	`TCP` | `SSL`	O		u	w	
	The broker transport may be specified as any combination of one or more of the following methods: `TCP` TCP/IP is supported. `SSL` SSL/TLS is supported. `NET` Entire Net-Work is supported. This value is not supported for a broker under Linux or Windows. **Examples:** `TRANSPORT=NET` specifies that only the Entire Net-Work transport method will be supported by the broker.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	TRANSPORT=TCP-NET specifies that both the TCP/IP and Net-Work transport methods will be supported by the broker. TRANSPORT=TCP-SSL-NET specifies that the TCP/IP, SSL/TLS, and Entire Net-Work transport methods will be supported by the broker. The parameters for each transport method are described in the respective section: **TCP** | **SSL** | **NET**.					
TRAP-ERROR	*nnnn*	O	z	u	w	b
	Where *nnnn* is the four-digit API error number that triggers the trace handler, for example 0007 (Service not registered). Leading zeros are not required. There is no default value. See *Deferred Tracing* under z/OS | Linux | Windows in the platform-specific Administration documentation.					
TRBUFNUM	*n*	O	z	u	w	b
	Changes the trace to write trace data to internal trace buffers. *n* is the size of the trace buffer in 64 KB units. There is no default value.					
TRMODE	WRAP	O	z	u	w	b
	Changes the trace mode. WRAP is the only possible value. This value instructs broker to write the trace buffer (see TRBUFNUM) if an event occurs. This event is triggered by a matching TRAP-ERROR during request processing or when an exception occurs.					
UMSG	See MAX-MESSAGES-IN-UOW.					
UOW-DATA-LIFETIME	1D | *n*S | *n*M | *n*H | *n*D	O	z	u	w	b
	Defines the default lifetime for units of work for the service. *n*S Number of seconds the UOW can exist (max. 2147483647). *n*M Number of minutes the UOW can exist (max. 35791394). *n*H Number of hours the UOW can exist (max. 596523). *n*D Number of days the UOW can exist (max. 24855). If the UOW is inactive - that is, is not processed within the time limit - it is deleted and given a status of TIMEOUT. This attribute can be overridden by the UWTIME field in the Broker ACI control block. See *Timeout Considerations for EntireX Broker*.					
UOW-MSGS	See MAX-MESSAGES-IN-UOW.					
UOW-STATUS-LIFETIME	no value | *n*[S] | *n*M | *n*H | *n*D	O	z	u	w	b
	The value to be added to the UOW-DATA-LIFETIME (lifetime of associated UOW). If a value is entered, it must be 1 or greater; a value of 0 will result in an error. If no value is entered, the lifetime of the UOW *status* information will be the same as the lifetime of the UOW itself.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	*n*S Number of seconds the UOW status exists longer than the UOW itself (max. 2147483647). *n*M Number of minutes (max. 35791394). *n*H Number of hours (max. 596523). *n*D Number of days (max. 24855). This attribute is ignored if PSTORE=NO is defined. The lifetime determines how much additional time the UOW status is retained in the persistent store and is calculated from the time at which the associated UOW enters any of the following statuses: PROCESSED, TIMEOUT, BACKEDOUT, CANCELLED, DISCARDED. The additional lifetime of the UOW status is calculated only when broker is executing. Value in UOW-STATUS-LIFETIME supersedes the value (if specified) in attribute UWSTATP. **Note:** If no unit is specified, the default unit is seconds. The unit does not have to be identical to the unit specified for UOW-DATA-LIFETIME.					
UWSTAT-LIFETIME	Alias for UOW-STATUS-LIFETIME.					
UWSTATP	0 | *n*	O	z	u	w	b
	Contains a multiplier used to compute the lifetime of a persistent status for the service. The UWSTATP value is multiplied by the UOW-DATA-LIFETIME value (the lifetime of the associated UOW) to determine the length of time the status will be retained in the persistent store. 0 The status is not persistent. 1-254 Multiplied by the value of UOW-DATA-LIFETIME to determine how long a persistent status will be retained. **Note:** This attribute has not been supported since EntireX version 7.3. Use UOW-STATUS-LIFETIME instead.					
UWTIME	Alias for UOW-DATA-LIFETIME.					
WAIT-FOR-ACTIVE-PSTORE	NO | YES	O	z	u	w	b
	Determines whether broker should wait for the Adabas Persistent Store to become active, or until c-tree PSTORE files become available. NO If broker should start with a PSTORE-TYPE=ADABAS and the database is not active or is not accessible, broker will stop. If broker should start with a PSTORE-TYPE=CTREE and the c-tree files are still in use, broker will stop. YES If broker should start with a PSTORE-TYPE=ADABAS and the database is not active or is not accessible, broker will retry every 10 seconds to initiate communications					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	with the PSTORE. Broker will reject any user requests until it is able to contact the Adabas database. If broker should start with a `PSTORE-TYPE=CTREE` and the c-tree files are still in use, broker will retry every 10 seconds to rebuild the persistent data. Broker will reject any user requests until it is able to rebuild the persistent data.					
`WORKER-MAX`	`64` | *n* (min. `1`, max. `64`)	O	z	u	w	b
	Maximum number of worker tasks the broker can use.					
`WORKER-MIN`	`1` | *n* (min. `1`, max. `64`)	O	z	u	w	b
	Minimum number of worker tasks the broker can use.					
`WORKER-NONACT`	`70S` | *n* | *n*`S` | *n*`M` | *n*`H`	O	z	u	w	b
	Non-activity time to elapse before a worker tasks is stopped. *n* Same as *n*`S`. *n*`S` Non-activity time in seconds (default `70`, max. `2147483647`). *n*`M` Non-activity time in in minutes (max. `35791394`). *n*`H` Non-activity time in hours (max. `596523`). **Caution:** A value of 0 (zero) is invalid. If you set this value too low, additional overhead is required for starting and stopping worker tasks. The default and recommended value is `70S`.					
`WORKER-QUEUE-DEPTH`	`1` | *n* (min. `1`)	O	z	u	w	b
	Number of unassigned user requests in the input queue before another worker task gets started. The default and recommended value is `1`. A higher value will result in longer broker response times.					
`WORKER-START-DELAY`	*`internal-value`* | *n*	O	z	u	w	b
	n Delay is extended by *n* seconds. Delay after a successful worker task invocation before another worker task can be started to handle current incoming workload. This attribute is used to avoid the risk of recursive invocation of worker tasks, because starting a worker task itself causes workload increase. If no value is specified, an internal value calculated by the broker is used to optimize dynamic worker management. This calculated value is the maximum time required to start a worker task.					

Service-specific Attributes

Each section begins with the keyword DEFAULTS=SERVICE. Services with common attribute values can be grouped together. The attributes defined in the grouping apply to all services specified within it. However, if a different attribute value is defined immediately following the service definition, that new value applies. See also the sections *Wildcard Service Definitions* and *Service Update Modes* below the table.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
APPLICATION-MONITORING or APPMON	YES | NO	O	z	u	w	b
	YES Enable application monitoring for the specified services. NO Disable application monitoring for the specified services. See the separate Application Monitoring documentation.					
APPLICATION-MONITORING-NAME or APPMON-NAME	A100	O	z	u	w	b
	Specifies the application monitoring name. Used to set the value of the **ApplicationName** KPI. If omitted, the default value from the APPLICATION-MONITORING section is used. If this value is also not specified, the corresponding CLASS/SERVER/SERVICE names are used. See the separate Application Monitoring documentation.					
CLASS	A32 (case-sensitive)	R	z	u	w	b
	Part of the name that identifies the service together with the SERVER and SERVICE attributes. CLASS must be specified first, followed immediately by SERVER and SERVICE. The following rules apply: ■ Classes starting with any of the following are reserved for use by Software AG. Do not use these in applications you write: BROKER, SAG, ENTIRE, ETB, RPC, ADABAS, NATURAL. ■ Valid characters for class name are letters a-z, A-Z, numbers 0-9, hyphen and underscore. ■ Do not use dollar, percent, period or comma. See also the restriction for SERVICE attribute names.					
CLIENT-RPC-AUTHORIZATION	N | Y	O	z			b
	Determines whether this service is subject to RPC authorization checking. N No RPC authorization checking is performed. Y RPC library and program name are appended to the authorization check performed by EntireX Security. Specify YES only to RPC-supported services.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	B
	To allow conformity with Natural Security, the `CLIENT-RPC-AUTHORIZATION` paramet optionally be defined with a prefix character as follows: `CLIENT-RPC-AUTHORIZATION=(YES,<prefix-character>)`.					
`CONV-LIMIT`	`UNLIM` | *n*	O	z	u	w	
	Allocates a number of conversations especially for this service. `UNLIM` The number of conversations is restricted only by the number of conversations gl available. Precludes the use of `NUM-CONVERSATION=AUTO` in the Broker section attribute file. *n* Number of conversations. A value of 0 (zero) is invalid. If `NUM-CONVERSATION=AUTO` is specified in the Broker section of the attribute file, `CONV-LIMIT=UNLIM` is not allowed in the service section. A value must be specified or t `CONV-LIMIT` attribute must be suppressed entirely for the service so that the default (`CONV-DEFAULT`) becomes active.					
`CONV-NONACT`	`5M` | *n* | *n*`S` | *n*`M` | *n*`H`	R	z	u	w	
	Non-activity time for connections. *n* Same as *n*`S`. *n*`S` Non-activity time in seconds (max. 2147483647). *n*`M` Non-activity time in minutes (max. 35791394). *n*`H` Non-activity time in hours (max. 596523). A value of 0 (zero) is invalid. If a connection is not used for the specified time, that is, a or a client does not issue a broker request that references the connection in any way, the connection is treated as inactive and the allocated resources are freed.					

`CONVERSION`	`A255`
	
	`(SAGTCHA[,TRACE=`*n*`][,`*OPTION*`=`*s*`]|`
	`SAGTRPC[,TRACE=`*n*`][,`*OPTION*`=`*s*`]|`
	name`[,TRACE=`*n*`]|`
	`NO)`
		O	z	u	w	

	Defines ICU conversion or SAGTRPC user exit for character conversion. See *Internationa with EntireX*. `SAGTCHA` (1) Conversion using ICU Conversion for *ACI-based Programming*. `SAGTRPC` (2) Conversion using ICU Conversion for *RPC-based Components* and *Reliable R* *name* (3) Name of the SAGTRPC user exit for RPC-based components and Reliable I See also *Configuring SAGTRPC User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000

	Writing SAGTRPC User Exits under *Configuring Broker for Internationalization* in the platform-specific Administration documentation.
	`NO` If conversion is not to be used, either omit the `CONVERSION` attribute or specify `CONVERSION=NO`, for example for binary payload.
	
	The `CONVERSION` attribute overrides the `TRANSLATION` attribute when defined for a service. That is, when `TRANSLATION` and `CONVERSION` are both defined, `TRANSLATION` will be ignored.
	
	Note:
	1. See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation.
	2. SAGTRPC is not supported on BS2000. For conversion with single-byte code pages, use SAGTCHA on BS2000 for *RPC-based Components* and *Reliable RPC*.
	3. SAGTRPC user exit is not supported on BS2000.
	
	`TRACE`
	
	If tracing is switched on, the trace output is written to the broker log file. The following trace levels are available:
	
	`0` No tracing
	`1 STANDARD` This level is an "on-error" trace. It provides information on conversion errors only. For RPC calls this includes the IDL library, IDL program and the data. Note that if ***`OPTION` Values for Conversion*** are set, errors are ignored.
	`2 ADVANCED` Tracing of incoming, outgoing parameters and the payload.
	`3 SUPPORT` This trace level is for support diagnostics. Use only when requested by Software AG Support.
	
	`OPTION`
	
	See table of possible values under ***`OPTION` Values for Conversion***.					

`DEFERRED`	`NO` | `YES`	O	z	u	w	b
	`NO` Units of work cannot be sent to the service until it is available. `YES` Units of work can be sent to a service that is not up and registered. The units of work will be processed when the service becomes available.					
`LOAD-BALANCING`	`YES` | `NO`	O	z	u	w	b
	`YES` When servers that offer a particular service are started, new conversations will be assigned to these servers in a round-robin fashion. The first waiting server will get the first new conversation, the second waiting server will get the second new conversation, and so on.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	B
	NO A new conversation is always assigned to the first server in the queue.					
LONG-BUFFER-LIMIT	UNLIM | *n*	O	z	u	w	
	Allocates a number of long message buffers for the service. UNLIM The number of long message buffers is restricted only by the number of buffers gl available. Precludes the use of NUM-LONG-BUFFER=AUTO in the Broker section o attribute file. *n* Number of long message buffers. A value of 0 (zero) is invalid. If NUM-LONG-BUFFER=AUTO is specified in the Broker secti the attribute file, LONG-BUFFER-LIMIT=UNLIM is not allowed in the service section. A v must be specified or the LONG-BUFFER-LIMIT attribute must be suppressed entirely fo service so that the default (LONG-BUFFER-DEFAULT) becomes active.					
MAX-MESSAGES-IN-UOW	16 | *n*	O	z	u	w	
	Maximum number of messages in a UOW.					
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	
	Maximum message size that can be sent to a service. This is transport-dependent. The default value represents the highest positive number t be stored in a four-byte integer.					
MAX-MSG	See MAX-MESSAGE-LENGTH.					
MAX-UOW-MESSAGE-LENGTH	See MAX-MESSAGE-LENGTH.					
MAX-UOWS	0 | *n*	O	z	u	w	
	0 The service does not accept units of work, that is, it processes only messages that are n of a UOW. Using zero prevents the sending of UOWs to services that are not intende process them. *n* Maximum number of UOWs that can be active concurrently for the service. If you do provide a MAX-UOWS value for the service, it defaults to the MAX-UOWS setting for the b If you provide a value that exceeds that of the broker, the service MAX-UOWS is set to t broker's MAX-UOWS value and a warning message is issued. Specify MAX-UOWS=0 for Natural RPC Servers. This restriction will be removed with a la release.					
MUOW	See MAX-UOWS.					
NOTIFY-EOC	NO | YES	O	z	u	w	
	Specifies whether timed-out conversations are to be stored or discarded. NO Discard the EOC notifications if the server is not ready to receive.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	`YES` Store the EOC notifications if the server is not ready to receive and then notify the server if possible. If a server is not ready to receive an EOC notification, it can be stored or discarded. If it is stored, the server is notified, if possible, when it is ready to receive. **Caution:** The behavior activated by this parameter can be relied upon only during a single lifetime of the broker kernel. Specifically, conversations containing units of work, whose lifetime can span multiple broker kernel sessions, cannot be assumed to show this behavior, even with `NOTIFY-EOC=YES`.					
`NUM-UOW`	Alias for `MAX-UOWS`.					
`POSTPONE-ATTEMPTS`	`0` | *n*	O	z	u	w	
	Defines the number of attempts putting a received unit of work (UOW) due to `SYNCPOINT` option `CANCEL` on the postponed queue for later processing. `0` All UOWs rejected by the receiver (`SYNCPOINT` option `CANCEL`) will be cancelled immediately. Attribute `POSTPONE-DELAY` is ignored. *n* Defines the number of postpone attempts that are performed instead of considering the UOW finished due to `SYNCPOINT` option `CANCEL`; the UOW will be moved to the postponed queue and the UOW status will be changed to `POSTPONED`. These UOWs will be delivered to the receiver when the time specified with `POSTPONE-DELAY` has elapsed. **Note:** Broker-specific attribute `POSTPONED-QUEUE` must be enabled (default) for this attribute to take effect. The default value is 0. See *Postponing Units of Work*.					
`POSTPONE-DELAY`	`0` | *n* | *n*`S` | *n*`M` | *n*`H`	O	z	u	w	
	The length of time a UOW is kept in status `POSTPONED`. `0` No postponed queue is created and attribute `POSTPONE-ATTEMPTS` is ignored. *n*`S` Number of seconds the UOW stays unreadable in the postponed queue with status `POSTPONED` (max. 2147483647). *n*`M` Number of minutes the UOW stays unreadable in the postponed queue with status `POSTPONED` (max. 35791394). *n*`H` Number of hours the UOW stays unreadable in the postponed queue with status `POSTPONED` (max. 596523). *n*`D` Number of days the UOW stays unreadable in the postponed queue with status `POSTPONED` (max. 24855). The status of the UOW will be changed from `POSTPONED` to `ACCEPTED` after elapsed `POSTPONE-DELAY`. This delay time does not affect the `UOW-DATA-LIFETIME`. The `POSTPONE-DELAY` must be less than `UOW-STATUS-LIFETIME` in order to make the UOW receivable again.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	B
	Note: Broker-specific attribute POSTPONED-QUEUE must be enabled (default) for this att to take effect. The default is 0, that is, no postponed queue is created, but if a value is en the minimum delay is 30 seconds. Any value entered that is less than 30 seconds will be inc to this value. See *Postponing Units of Work*.					
SERVER	A32 (case-sensitive)	R	z	u	w	
	Part of the name that identifies the service together with the CLASS and SERVICE attribu CLASS must be specified first, followed immediately by SERVER and SERVICE. Valid characters for server name are letters a-z, A-Z, numbers 0-9, hyphen and undersco not use dollar, percent, period or comma.					
SERVER-DEFAULT	*n* | UNLIM	O	z	u	w	
	Default number of servers that are allowed for every service. *n* Number of servers. UNLIM The number of servers is restricted only by the number of servers globally avail Precludes the use of NUM-SERVER=AUTO. A value of 0 (zero) is invalid. This value can be overridden by specifying a SERVER-LIMIT for the service.					
SERVER-LIMIT	*n* | UNLIM	O	z	u	w	
	Allows a number of servers especially for this service. *n* Number of servers. UNLIM The number of servers is restricted only by the number of servers globally avail Precludes the use of NUM-SERVER=AUTO in the Broker section of the attribute fil A value of 0 (zero) is invalid. If NUM-SERVER=AUTO is specified in the Broker section of the attribute file, SERVER-LIMIT= is not allowed in the service section. A value must be specified or the SERVER-LIMIT at must be suppressed entirely for the service so that the default (SERVER-DEFAULT) beco active. **Note:** Linux and Windows: This limit also includes any attach server you are using. Mal you increase the number by one for each attach server you use.					
SERVER-NONACT	5M | *n* | *n*S | *n*M | *n*H	R	z	u	w	
	Non-activity time for servers. A server that does not issue a broker request within the sp time limit is treated as inactive and all resources for the server are freed. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647).					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	*n*M Non-activity time in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). If a server registers multiple services, the highest value of all the services registered is taken as non-activity time for the server.					
SERVICE	A32 (case-sensitive)	R	z	u	w	b
	Part of the name that identifies the service together with the CLASS and SERVER attributes. CLASS must be specified first, followed immediately by SERVER and SERVICE. The SERVICE attribute names EXTRACTOR and DEPLOYMENT are reserved for Software AG internal use and should not be used in customer-written applications. Valid characters for service name are letters a-z, A-Z, numbers 0-9, hyphen and underscore. Do not use dollar, percent, period or comma. See also the restriction for CLASS attribute names.					
SHORT-BUFFER-LIMIT	UNLIM | *n*	O	z	u	w	b
	Allocates a number of short message buffers for the service. UNLIM The number of short message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM-SHORT-BUFFER=AUTO in the Broker section of the attribute file. *n* Number of short message buffers. If NUM-SHORT-BUFFER=AUTO is specified in the Broker section of the attribute file, SHORT-BUFFER-LIMIT=UNLIM is not allowed in the service section. A value must be specified or the SHORT-BUFFER-LIMIT attribute must be suppressed entirely for the service so that the default (SHORT-BUFFER-DEFAULT) becomes active.					
STORE	OFF | BROKER	O	z	u	w	b
	Sets the default STORE attribute for all units of work sent to the service. OFF Units of work are not persistent. BROKER Units of work are persistent. This attribute can be overridden by the STORE field in the Broker ACI control block.					
TRANSLATION	NO | *name* (A255)	O	z	u	w	b
	Activates translation user exit for character conversion. NO If translation is not to be used - for example for binary payload (broker messages) - either omit the TRANSLATION attribute or specify TRANSLATION=NO. *name* Name of Translation User Exit. See also *Configuring Translation User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	B
	documentation or *Writing Translation User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. The CONVERSION attribute overrides the TRANSLATION attribute when defined for a ser that is, when TRANSLATION and CONVERSION are both defined, TRANSLATION will be ig					
UMSG	Alias for MAX-MESSAGES-IN-UOW.					
UOW-DATA-LIFETIME	1D | *n*S | *n*M | *n*H | *n*D	O	z	u	w	
	Defines the default lifetime for units of work for the service. *n*S Number of seconds the UOW can exist (max. 2147483647). *n*M Number of minutes the UOW can exist (max. 35791394). *n*H Number of hours the UOW can exist (max. 596523). *n*D Number of days the UOW can exist (max. 24855). This attribute is ignored if PSTORE=NO is defined. If the unit of work (UOW) is inactive, that is, not processed within the time limit, it is de and given a status of TIMEOUT. This attribute can be overridden by the UWTIME field in t Broker ACI control block.					
UOW-MSGS	Alias for MAX-MESSAGES-IN-UOW.					
UOW-STATUS-LIFETIME	no value | *n*[S] | *n*M | *n*H | *n*D	O	z	u	w	
	The value to be added to the UOW-DATA-LIFETIME lifetime of associated UOW). If a val entered, it must be 1 or greater; a value of 0 will result in an error. If no value is entered, lifetime of the UOW *status* information will be the same as the lifetime of the UOW itsel *n*S Number of seconds the UOW status exists longer than the UOW itself (max. 214748 *n*M Number of minutes (max. 35791394). *n*H Number of hours (max. 596523). *n*D Number of days (max. 24855). The lifetime determines how much additional time the UOW status is retained in the per store and is calculated from the time at which the associated UOW enters any of the fol statuses: PROCESSED, TIMEOUT, BACKEDOUT, CANCELLED, DISCARDED. The additional lif of the UOW status is calculated only when broker is executing. Value in UOW-STATUS-LIF supersedes the value (if specified) in attribute UWSTATP. **Note:** If no unit is specified, the default unit is seconds. The unit does not have to be ide to the unit specified for UOW-DATA-LIFETIME.					
UWSTATP	0 | *n*	O	z	u	w	

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Contains a multiplier used to compute the lifetime of a persistent status for the service. The UWSTATP value is multiplied by the UOW-STATUS-LIFETIME value (the lifetime of the associated UOW) to determine the length of time the status will be retained in the persistent store. 0 The status is not persistent. 1 - 254 Multiplied by the value of UOW-DATA-LIFETIME to determine how long a persistent status will be retained. This attribute is ignored if PSTORE=NO is defined. **Note:** This attribute has not been supported since EntireX version 7.3. Use UOW-STATUS-LIFETIME instead.					
UWSTAT-LIFETIME	Alias for UOW-STATUS-LIFETIME.					
UWTIME	Alias for UOW-DATA-LIFETIME.					

Wildcard Service Definitions

The special names of CLASS = *, SERVER = * and SERVICE = * are allowed in the service-specific and authorization rule-specific sections of the broker attribute file. These are known as "wildcard" service definitions. If this name is present in the attribute file, any service that registers with the broker and does not have its own entry in the attribute file will inherit the attributes that apply to the first wildcard service definition found.

For example, a server that registers with CLASS=ACLASS, SERVER=ASERVER and SERVICE=ASERVICE can inherit attributes from any of the following entries in the attribute file (this list is not necessarily complete):

```
CLASS = *, SERVER = ASERVER, SERVICE = ASERVICE
CLASS = ACLASS, SERVER = *, SERVICE = *
CLASS = *, SERVER = *, SERVICE = *
```

Of course, if there is a set of attributes that are specifically defined for CLASS=ACLASS, SERVER=ASERVER, SERVICE=ASERVICE, then all of the wildcard service definitions will be ignored in favor of the exact matching definition.

Service Update Modes

EntireX has two modes for handling service-specific attributes. See broker-specific attribute SERVICE-UPDATES.

- In **service update mode** (SERVICE-UPDATES=YES), the service configuration sections of the attribute file are read whenever the first replica of a particular service registers.
- In **non-update mode** (SERVICE-UPDATES=NO), the attribute file is not reread. All attributes are read during startup and the broker does not honor any changes in the attribute file. This mode is useful if
  - there is a high frequency of REGISTER operations, or
  - the attribute file is rather large and results in a high I/O rate for the broker.

  The disadvantage to using non-update mode is that if specific attributes are modified, the broker must be restarted to effect the changes. Generally, this mode should be used only if the I/O rate of the broker is considerably high, and if the environment seldom changes.

OPTION Values for Conversion

The different option values allow you to either handle character conversion deficiencies as errors, or to ignore them:

1. Do not ignore any character conversion errors and force an error always (value STOP). This is the default behavior.
2. Ignore if characters cannot be converted into the receiver's codepage, but force an error if sender characters do not match the sender's codepage (value SUBSTITUTE-NONCONV).
3. Ignore any character conversion errors (values SUBSTITUTE and BLANKOUT).

Situations 1 and 2 above are reported to the broker log file if the TRACE option for CONVERSION is set to level 1.

Value	Description	Options Supported for		Report Situation in Broker Log File if TRACE Option for CONVERSION is set to 1	
		SAGTCHA	SAGTRPC	Bad Input Characters (Sender's Codepage)	Non-convertible Characters (Receiver's Codepage)
SUBSTITUTE	Substitutes both non-convertible characters (receiver's codepage) and bad input characters (sender's codepage) with a	YES	YES	No message.	No message

Value	Description	Options Supported for		Report Situation in Broker Log File if TRACE Option for CONVERSION is set to 1	
		SAGTCHA	SAGTRPC	Bad Input Characters (Sender's Codepage)	Non-convertible Characters (Receiver's Codepage)
	codepage-dependent default replacement character.				
SUBSTITUTE-NONCONV	If a corresponding code point is not available in the receiver's codepage, the character cannot be converted and is substituted with a codepage-dependent default replacement character. Bad input characters in sender's codepage are not substituted and result in an error.	YES	YES	Write detailed conversion error message.	No message.
BLANKOUT	Substitutes non-convertible characters with a codepage-dependent default replacement; blanks out the complete RPC IDL field containing one or more bad input characters.	NO	YES	No message.	No message.
STOP	Signals an error on detecting a non-convertible or bad input character. This is the default behavior if no option is specified.	YES	YES	Write detailed conversion error message.	Write detailed conversion error message.

Codepage-specific Attributes

The codepage-specific attribute section begins with the keyword `DEFAULTS=CODEPAGE` as shown in the sample attribute file. You can use the attributes in this section to customize the broker's locale string defaults and customize the mapping of locale strings to codepages for character conversion with ICU conversion and SAGTRPC user exit. See *Internationalization with EntireX* for more information.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`DEFAULT_ASCII`	Any ICU converter name or alias. See also ***Additional Notes*** below.	O	z	u	w	b

	Customize the broker's locale string defaults by assigning the default codepage for EntireX components (client or server). See *Broker's Locale String Defaults*. This value is used instead of the broker's locale string defaults if
	■ the calling component does not send a locale string itself, and
	■ the calling component is running on an ASCII platform (Linux, Windows, etc.)
	Example:
	`DEFAULTS=CODEPAGE`
	`  * Broker Locale String Defaults`
	`  DEFAULT_ASCII=windows-950`
	For more examples, see *Configuring Broker's Locale String Defaults* in the Internationalization documentation and also ***Additional Notes*** below.					

`DEFAULT_EBCDIC_IBM`	Any ICU converter name or alias	O	z	u	w	b
	Customize the broker's locale string defaults by assigning the default codepage for EntireX components (client or server). See *Broker's Locale String Defaults*. This value is used instead of the broker's locale string defaults if ■ the calling component does not send a locale string itself and ■ the calling component is running on an IBM mainframe platform Example:					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000

	`DEFAULT=CODEPAGE`
	`  DEFAULT_EBCDIC_IBM=ibm-937`
	
	For more examples, see *Configuring Broker's Locale String Defaults* in the Internationalization documentation and also ***Additional Notes*** below.					

`DEFAULT_EBCDIC_SNI`	Any ICU converter name or alias.	O	z	u	w	b

	Customize the broker's locale string defaults by assigning the default codepage for EntireX components (client or server). See *Broker's Locale String Defaults*. This value is used instead of the locale string defaults if
	
	■ the calling component does not send a locale string itself, and
	■ the calling component is running on a Fujitsu EBCDIC mainframe platform (BS2000)
	
	Example:
	
	`DEFAULT=CODEPAGE`
	`  DEFAULT_EBCDIC_SNI= bs2000-edf03drv`
	
	For more examples, see *Configuring Broker's Locale String Defaults* in the Internationalization documentation and also ***Additional Notes*** below.					

`locale-string`	Any ICU converter name or alias. See also ***Additional Notes*** below.	O	z	u	w	
	Customize the mapping of locale strings to codepages and bypass the broker's locale string processing mechanism. See *Broker's Locale String Processing*. This is useful: ■ if the broker's locale string processing fails - that is, it leads to no codepage or to the wrong codepage - you can explicitly assign the codepage which meets your requirements. ■ if you want to install user-written ICU converters (codepages) into the broker, see *Building and Installing ICU Custom Converters* in the platform-specific Administration documentation. The attribute (locale string) is the locale string sent by your EntireX component (client or server) and the value is the codepage that you want to use in place of that locale string. In the first line of the example below, the client or server application sends ASCII as a locale string; the broker maps this to the codepage ISO 8859_1. In the same way EUC_JP_LINUX is mapped to ibm-33722_P12A-1999. All other locale strings are mapped by the broker's mapping mechanism, see *Broker's Built-in Locale String Mapping*. Example:					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000

	`DEFAULTS=CODEPAGE`
	`  * Broker Locale String Codepage Assignments`
	`  ASCII=ISO8859`
	`  EUC_JP_LINUX=ibm-33722_P12A-1999`
	`  * Customer-written ICU converters`
	`  CP1140=myebcdic`
	`  CP0819=myascii`
	
	For more examples, see *Bypassing Broker's Built-in Locale String Mapping* and also ***Additional Notes*** below.					

Additional Notes

- Locale string matching is case insensitive when bypassing the broker's built-in mechanism, that is, when the broker examines the codepages section in the attribute file.
- If ICU is used for character conversion and the style in not known by ICU, e.g. <ll>_<cc> etc., the name will be mapped to a suitable ICU alias. For more details on the mapping mechanism, see *Broker's Built-in Locale String Mapping*. For more details on ICU and ICU converter name standards, see *ICU Resources*.
- If SAGTRPC user exit is used for the character conversion, we recommend assigning the codepage in the form CP<*nnnnn*>. To determine the number given to SAGTRPC user exit, see *Broker's Built-in Locale String Mapping*.
- See `CONVERSION` on this page for the character conversion in use.

Adabas SVC/Entire Net-Work-specific Attributes

The Adabas SVC/Entire Net-Work-specific attribute section begins with the keyword `DEFAULTS=NET` as shown in the sample attribute file. The attributes in this section are needed to execute the Adabas SVC/Entire Net-Work communicator of the EntireX Broker kernel.

Note: This section applies to mainframe platforms only. It does not apply to Linux and Windows.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`ADASVC`	*nnn*	R	z			
	Sets the Adabas SVC number for EntireX Broker access. The Adabas SVC is used to perform various internal functions, including communication between the caller program and EntireX Broker. Not supported on BS2000.					
`EXTENDED-ACB-SUPPORT`	`NO` | `YES`	O	z			b
	Determines whether extended features of Adabas version 8 (or above) are supported. `NO` No features of Adabas version 8 or above will be used. `YES` Informs broker kernel to provide Adabas/WAL version 8 transport capability. This parameter is required for sending/receiving more than 32 KB data over Adabas [NET] transport. This value should be set only if you have installed Adabas/WAL version 8, Adabas SVC, and included Adabas/WAL version 8 load libraries into the steplib of broker kernel; otherwise, unpredictable results can occur.					
`FORCE`	`NO` | `YES`	O	z			b
	Determines whether DBID table entries can be overwritten. `NO` Overwrite of DBID table entries not permitted. `YES` Overwrite of DBID table entries permitted. This is required when the DBID table entry is not deleted after abnormal termination. **Caution:** Overwriting an existing entry prevents any further communication with the overwritten node. Use `FORCE=YES` only if you are absolutely sure that no target node with that DBID is active.					
`IDTNAME`	*idtname*`(A8)` | `ADABAS5B`	O				b
	If an ID table name is specified with the appropriate `ADARUN` parameter for Entire Net-Work, Adabas or Natural, the same name must be specified here.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	The ID table is used to perform various internal functions, including communication between the caller program and the EntireX Broker. Only supported under BS2000.					
IUBL	8000 | *n*	O	z			b
	This parameter sets the maximum length (in bytes) of the buffer that can be passed from the caller to EntireX Broker. The maximum size of IUBL is the same as the maximum value of the Adabas parameter LU. See the *Adabas Operations Manual*. IUBL must be large enough to hold the maximum send-length plus receive-length required for any caller program plus any administrative overhead for Adabas and Entire Net-Work control structures.					
LOCAL	NO | YES	O	z			b
	For remote nodes accessed via Entire Net-Work, the attribute LOCAL specifies whether the target ID defined with the NODE attribute can be accessed only locally, or also remotely. NO DBID is *global* and can be accessed from remote nodes via Entire Net-Work. YES DBID is *local* and cannot be accessed from remote nodes via Entire Net-Work.					
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	b
	Maximum message size that the broker kernel can process using transport method NET. The default value represents the highest positive number that can be stored in a four-byte integer.					
NABS	10 | *n*	O	z			b
	The number of attached buffers to be used (max. 524287). An attached buffer is an internal buffer used for interprocess communication. An attached buffer pool equal to the NABS value multiplied by 4096 will be allocated. This buffer pool must be large enough to hold all data (IUBL) of all parallel calls to EntireX Broker. The following formula can be used to calculate the value for NABS: NABS = NCQE *IUBL / 4096.					
NCQE	10 | *n*	O	z			b
	NCQE defines the number of command queue elements which are available for processing commands arriving at the broker kernel over Adabas SVC / Net-Work transport mechanism. Sufficient NCQE should be allocated to allow this transport mechanism to process multiple broker commands concurrently. Each command queue element requires 192 bytes, and the element is released when either the user (client or server) has received the results of the command, or if the command is timed out.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	The number of command queue elements required to handle broker calls depends on the number of parallel active broker calls that are using the transport mechanism Adabas SVC / Entire Net-Work. For example, all broker commands issued by client or server components using this transport mechanism:					
NODE	1-65534	R	z			b
	Defines the unique DBID for EntireX Broker. Used for internode Adabas/Entire Net-Work communication. There is no default; the value of NODE must be a value greater than or equal to 1 or less than or equal to 65534. If you set the parameter LOCAL=YES, you can use the same node number for different installations of EntireX Broker in an Entire Net-Work environment.					
TIME	30 | *n*	O	z			b
	This parameter sets the timeout value for broker calls in seconds. The results of a broker call must be received by the caller within this time limit.					
TRACE-LEVEL	0-4	O	z			b
	The level of tracing to be performed while the broker is running with transport method NET. It overrides the global value of trace level for all NET routines. 0 No tracing. Default value. 1 Display invalid Adabas commands. 2 All of trace level 1, plus errors if request entries could not be allocated. 3 All of trace level 2, plus all routines executed. 4 All of trace level 3, plus function arguments and return values. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.					

Security-specific Attributes

The security-specific attribute section begins with the keyword `DEFAULTS=SECURITY` as shown in the sample attribute file. This section applies only if broker-specific attribute `SECURITY=YES` is specified.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`ACCESS-SECURITY-SERVER`	`NO` | `YES`	O				b
	Determines where authentication is checked. `NO` Authentication is checked in the broker tasks. This requires broker to be running under TSOS in order to execute privileged security checks. `YES` Authentication is checked in the EntireX Broker Security Server for BS2000. This does not require broker to be running under TSOS. See *EntireX Broker Security Server for BS2000*.					
`APPLICATION-NAME`	A8	O	z			

	Specifies the name of the application to be checked if `FACILITY-CHECK=YES` is defined. In RACF, for example, an application `BROKER` with read permission for user `DOE` is defined with following commands:
	
	`RDEFINE APPL BROKER UACC(NONE)`
	`PERMIT BROKER CLASS(APPL) ID(DOE) ACCESS(READ)`
	`SETROPTS CLASSACT(APPL)`
	
	See attribute `FACILITY-CHECK` for more information.					

`AUTHORIZATION-DEFAULT`	`YES` | `NO`	O		u	w	
	Determines whether access is granted to a specified service if the specified service could not be found listed in the repository of authorization rules or in section *`DEFAULTS=AUTHORIZATION-RULES`* of the attribute file. `YES` Grant access. `NO` Deny access. Applies only when using EntireX Security under Linux and Windows. Authorization rules can be stored within a repository. When an authorization call occurs, EntireX Security uses the values of this parameter to perform an access check for a particular broker instance against an (authenticated) user ID and list of rules. See also *Authorization Rules*.					
`CHECK-IP-ADDRESS`	`YES` | `NO`	O	z			
	Determines whether the TCP/IP address of the caller is subject to a resource check.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
ERRTXT-MODULE	NA2MSG0 | NA2MSG1 | NA2MSG2 | *ModuleName*	O	z			
	Specifies the name of the security error text module. Default is NA2MSG0, English messages. For instructions on how to customize messages, see *Build Language-specific Messages (Optional)* under *Installing EntireX Security under z/OS*.					
FACILITY-CHECK	NO | YES	O	z			
	It is possible to check whether a particular user is at all allowed to use an application before performing a password check. The advantage of this additional check is that when the user is not allowed to use this application, the broker returns error 00080013 and does not try to authenticate the user. Failing an authentication check may lead to the user's password being revoked; this situation is avoided if the facility check is performed first. See attribute APPLICATION-NAME for further details. **Note:** This facility check is an additional call to the security subsystem and is executed before each authentication call.					
IGNORE-STOKEN	NO | YES	O	z	u	w	b
	Determines whether the value of the ACI field SECURITY-TOKEN is verified on each call.					
INCLUDE-CLASS	YES | NO	O	z			
	Determines whether the class name is included in the resource check.					
INCLUDE-NAME	YES | NO	O	z			
	Determines whether the server name is included in the resource check.					
INCLUDE-SERVICE	YES | NO	O	z			
	Determines whether the service name is included in the resource check.					
LDAP-AUTHENTICATION-URL	*ldapUrl*	O		u	w	
	Authentication is performed against the LDAP repository specified under *ldapUrl*. ■ **TCP** Specify repository URL: `LDAP-AUTHENTICATION-URL="ldap://HostName[:PortNumber]"` ■ **SSL/TLS** Specify repository URL with ldaps:					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000

	`LDAP-AUTHENTICATION-URL="ldaps://HostName[:PortNumber]"`
	
	If no port number is specified, the default is the standard LDAP port number 389 for TCP transport. Examples for TCP and SSL/TLS:
	
	`LDAP-AUTHENTICATION-URL="ldap://myhost.mydomain.com"`
	`LDAP-AUTHENTICATION-URL="ldaps://myhost.mydomain.com:636"`					

`LDAP-AUTHORIZATION-URL`	*`ldapUrl`*	O		u	w	

	Authorization is performed against the LDAP repository specified under *`ldapUrl`*.
	
	■ **TCP**
	Specify repository URL:
	
	`LDAP-AUTHORIZATION-URL="ldap://HostName[:PortNumber]"`
	
	If no port number is specified, the default is the standard LDAP port number 389 for TCP transport.
	Example for TCP:
	
	`LDAP-AUTHORIZATION-URL="ldap://myhost.mydomain.com:389"`
	
	This attribute replaces the parameters `host`, `port` and `protocol` in the *xds.ini* file of EntireX version 9.10 and below.					

`LDAP-AUTH-DN`	*`authDN`*	O		u	w	
	For authenticated access to the LDAP server. Specifies the DN of the user. Default value: `cn=admin,dc=software-ag,dc=de` This attribute replaces parameter `authDN` in the *xds.ini* file of EntireX version 9.10 and below.					
`LDAP-AUTH-PASSWD-ENCRYPTED`	*`authPass`*	O		u	w	
	For authenticated access to the LDAP server. Specifies the encrypted value of the user password. Use program `etbnattr` to get the encrypted password: `etbnattr -x clear_text_password -echo_password_only` This writes the encrypted password to standard output. This attribute replaces parameter `authPass` in the *xds.ini* file of EntireX version 9.10 and below.					
`LDAP-AUTHORIZATION-RULE`	`A32`	O		u	w	

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	List of authorization rules. Multiple sets of rules can be defined, each set is limited to 32 chars. The maximum number of `LDAP-AUTHORIZATION-RULE` entries in the attribute file is 16. Applies only when using EntireX Security under Linux or Windows and `SECURITY-SYSTEM=ldapUrl`. Authorization rules can be stored in an LDAP repository. When an authorization call occurs, EntireX Security uses the values of this parameter and `AUTHORIZATION-DEFAULT` to perform an access check for a particular broker instance against an (authenticated) user ID and list of rules. See also *Authorization Rules*.					
`LDAP-BASE-DN`	*baseDN*	O		u	w	
	Specifies the base distinguished name of the directory object that is the root of all objects for authorization rules. Default value: `dc=software-ag,dc=de` This attribute replaces parameter `baseDN` in the *xds.ini* file of EntireX version 9.10 and below.					
`LDAP-PERSON-BASE-BINDDN`	*ldapDn*	O		u	w	
	Used with LDAP authentication to specify the distinguished name where authentication information is stored. This value is prefixed with the user ID field name (see below). Example: `LDAP-PERSON-BASE-BINDDN="cn=users,dc=mydomain,dc=com"`					
`LDAP-REPOSITORY-TYPE`	`OpenLDAP` | `ActiveDirectory` | `SunOneDirectory` | `Tivoli` | `Novell` | `ApacheDS`	O		u	w	
	Use predefined known fields for the respective repository type. Specify the repository type that most closely matches your actual repository. In the case of Windows Active Directory, the user ID is typically in the form *domainName\userId*.					
`LDAP-SASL-AUTHENTICATION`	`NO` | `YES`	O			w	
	Specifies whether or not Simple Authentication and Security Layer (SASL) is to perform the authentication check. In practice, this determines whether or not the password supplied by the user is passed in plain text between the broker kernel and the LDAP server. If SASL is activated, this implies that the password is encrypted. `NO` Password is sent to LDAP server in plain text. `YES` Password is sent to LDAP server encrypted.					
`LDAP-USERID-FIELD`	`cn` | *uidFieldName*	O		u	w	

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Used with LDAP authentication to specify the first field name of a user in the Distinguished Name, for example: `LDAP-USERID-FIELD=`*`uid`					
`MAX-SAF-PROF-LENGTH`	`1-256`	O	z			
	This parameter should be increased if the length of the resource checks - that is, the length of the profile comprising "<class>.<server>.<service>" - is greater than 80 bytes. This parameter defaults to 80 if a value is not specified.					
`PASSWORD-TO-UPPER-CASE`	`NO` | `YES`	O	z			
	Determines whether the password and new password are converted to uppercase before verification.					
`PRODUCT`	`RACF` | `ACF2` | `TOP-SECRET`	O	z			
	Specifies the name of the installed security product. This attribute is used to analyze security-system-specific errors. The following systems are currently supported: `RACF` Security system RACF is installed. Default. `ACF2` Security system ACF2 is installed. `TOP-SECRET` Security system CA Top Secret is installed. The default value is used if an incorrect or no value is specified.					
`PROPAGATE-TRUSTED-USERID`	`YES` | `NO`	O	z			
	Determines whether a client user ID obtained by means of the trusted user ID mechanism is propagated to a server using the ACI field `CLIENT-USERID`.					
`SAF-CLASS`	`NBKSAG` | *`SAFClassName`*	O	z			
	Specifies the name of the SAF class/type used to hold the EntireX-related resource profiles.					
`SAF-CLASS-IP`	`NBKSAG` | *`SAFClassName`*	O	z			
	Specifies the name of the SAF class/type used when performing IP address authorization checks.					
`SECURITY-LEVEL`	`AUTHORIZATION` | `AUTHENTICATION`	O	z	u	w	b
	Specifies the mode of operation. `AUTHORIZATION` Authorization and authentication (not under BS2000). `AUTHENTICATION` Authentication. **Note:** In version 8.0, the default value for this parameter was `AUTHORIZATION`.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
SECURITY-NODE	YES | *name*	O	z			
	This parameter can be used to specify a prefix that is added to all authorization checks, enabling different broker kernels, in different environments, to perform separate authorization checks according to each broker kernel. For example, it is often important to distinguish between production, test, and development environments. YES This causes the broker ID to be used as a prefix for all authorization checks. *name* This causes the actual text (max. 8 chars) to be prefixed onto all authorization checks. **Note:** By *not* setting this parameter, no prefix is added to the resource check (the default behavior).					
SECURITY-SYSTEM	OS | LDAP	O	z	u	w	b
	OS Authentication is performed against the local operating system. Default if SECURITY=YES is specified and section DEFAULTS=SECURITY is omitted from the attribute file. LDAP Authentication and authorization are performed against the LDAP repository specified under LDAP-AUTHENTICATION-URL and LDAP-AUTHORIZATION-URL(not applicable to z/OS and BS2000).					
TRACE-LEVEL	0-4	O	z	u	w	b
	Trace level for EntireX Security. It overrides the global value of trace level in the attribute file. 0 No tracing. Default value. 1 Log security violations and access denied/permitted. 2 All of trace level 1, plus internal errors. 3 All of trace level 2, plus function entered/exit messages with argument values and some progress messages. 4 All of trace level 3, plus some selected data areas for problem analysis. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD. **Note:** Setting this value also affects tracing for authorization rules.					
TRUSTED-USERID	YES | NO	O	z			
	Activates the trusted user ID mechanism for broker requests arriving over the local Adabas IPC mechanism.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
USERID-TO-UPPER-CASE	NO | YES	O	z			
	Determines whether user ID is converted to uppercase before verification.					
UNIVERSAL	NO | YES	O	z			
	Determines whether access to undefined resource profiles is allowed.					
WARN-MODE	NO | YES	O	z	u	w	b
	Determines whether a resource check failure results in just a warning or an error.					

TCP/IP-specific Attributes

The TCP/IP-specific attribute section begins with the keyword `DEFAULTS=TCP` as shown in the sample attribute file. It contains attributes that apply to the TCP/IP transport communicator. The transport is activated by `TRANSPORT=TCP` in the Broker-specific section of the attribute file. A maximum of five TCP/IP communicators can be activated by specifying up to five `HOST/PORT` pairs.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`CERT-AUTHENTICATION`	`NO` | `YES`	O	z			
	`NO` Do not use SSL certificates for authentication. `YES` Use corresponding port for certificate-based authentication. See *Using SSL Certificates for Authentication* in the EntireX Security documentation for z/OS.					
`CONNECTION-NONACT`	*n* | *n*`S` | *n*`M` | *n*`H`	O	z	u	w	b
	Non-activity of the TCP/IP connection, after which a close is performed and the connection resources are freed. If this parameter is not specified here, broker will close the connection only when the application (or the network itself) terminates the connection. *n* Same as *n*`S`. *n*`S` Non-activity time in seconds (min. 600, max. 2147483647). *n*`M` Non-activity time in minutes (min. 10, max. 35791394). *n*`H` Non-activity time in hours (max. 596523). If not specified, the connection non-activity is not monitored by the broker. On the stub side, non-activity can be set with the environment variable `ETB_NONACT`. See *Limiting the TCP/IP Connection Lifetime* under z/OS | Linux | Windows | z/VSE in the platform-specific *Administering Broker Stubs* documentation. Both sides (the stub side and the broker kernel side) act independently of each other.					
`HOST`	`0.0.0.0` | *hostname* | *IP address*	O	z	u	w	b
	The address of the network interface on which broker will listen for connection requests. If `HOST` is not specified, broker will listen on any attached interface adapter of the system (or stack). A maximum of five `HOST/PORT` pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.					
`MAX-MESSAGE-LENGTH`	`2147483647` | *n*	O	z	u	w	b

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Maximum message size that the broker kernel can process using transport method TCP/IP. The default value represents the highest positive number that can be stored in a four-byte integer.					
PORT	1025-65535	O	z	u	w	b

	The TCP/IP port number on which the broker will listen for connection requests.
	
	If not specified, the broker will attempt to find its TCP/IP port number from the TCP/IP services file, using `getservbyname`. If it cannot find the number here, the default value of 1971 is used.
	
	A maximum of five `HOST/PORT` pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.
	
	Example for multiple ports on z/OS:
	
	`HOST=localhost,PORT=3930`
	`HOST=0.0.0.0,PORT=3931`
	
	■ Port 3930 is used for *local* TCP/IP communication only and is not visible outside the z/OS host.
	■ Port 3931 is used for *global* TCP/IP communication. With IBM's AT-TLS this port is turned into a TLS port, see *Running Broker with SSL/TLS Transport* in the z/OS Administration documentation.
	
	With this configuration you can reach the broker from outside the z/OS host via the secure TLS connection only (port 3931). The TCP connection (port 3930) can only be used from inside the z/OS host.					

RESTART	YES | NO	O	z	u	w	b
	`YES` The broker kernel will attempt to restart the TCP/IP communicator. `NO` The broker kernel will not try to restart the TCP/IP communicator. This setting applies to all TCP/IP communicators.					
RETRY-LIMIT	20 | *n* | UNLIM	O	z	u	w	b
	Maximum number of attempts to restart the TCP/IP communicator. This setting applies to all TCP/IP communicators.					
RETRY-TIME	3M | *n* | *n*S | *n*M | *n*H	O	z	u	w	b
	Wait time between stopping the TCP/IP communicator due to an unrecoverable error and the next attempt to restart it. *n* Same as *n*S. *n*S Wait time in seconds (max. 2147483647). *n*M Wait time in minutes (max. 35791394).					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	*n*H Wait time in hours (max. 596523). Minimum wait time is 1S. This setting applies to all TCP/IP communicators.					
REUSE-ADDRESS	YES | NO	O	z	u		b
	YES | NO	O			w	
	YES The TCP port assigned to the broker can be taken over and assigned to other applications (this is the default value on all non-Windows platforms). NO The TCP port assigned to the broker cannot be taken over and assigned to other applications. This is the default setting on Windows, and we strongly advise you do not change this value on this platform. **Note:** This setting might be required at your site when restarting broker immediately after stopping it. This is due to the inherent latency of the TCP/IP stack when closing connections.					
STACK-NAME	*StackName*	O	z			
	Name of the TCP/IP stack that the broker is using. If not specified, broker will connect to the default TCP/IP stack running on the machine.					
TRACE-LEVEL	0-4	O	z	u	w	b
	The level of tracing to be performed while the broker is running with transport method TCP/IP. It overrides the global value of trace level for all TCP/IP routines. 0 No tracing. Default value. 1 Display IP address of incoming request, display error number of outgoing error responses. 2 All of trace level 1, plus errors if request entries could not be allocated. 3 All of trace level 2, plus all routines executed. 4 All of trace level 3, plus function arguments and return values. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.					

c-tree-specific Attributes

The c-tree-specific attribute section begins with the keyword `DEFAULTS = CTREE`. The attributes in this section are optional. This section applies only if `PSTORE-TYPE = CTREE` is specified.

Not available under z/OS or BS2000.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
COMPATIBILITY	NO | YES	O		u	w	
	Determines whether the following c-tree parameters are set: ■ COMPATIBILITY PREV610A_FLUSH ■ COMPATIBILITY FDATASYNC ■ SUPPRESS_LOG_FLUSH YES ■ PREIMAGE_DUMP YES See your FairCom documentation for a description of these parameters. NO The c-tree parameters listed above are not set. Default. YES The c-tree parameters listed above are set. This provides compatibility with c-tree behavior prior to EntireX Broker 10.5.					
FLUSH-DIR	YES | NO	O		u	w	
	Controls whether metadata is flushed to disk immediately after creates, renames, and deletes of transaction log files and transaction-dependent files. YES Metadata is flushed to disk. NO Metadata is not flushed to disk. This provides compatibility with c-tree behavior prior to EntireX Broker version 10.5. See COMPATIBILITY NO_FLUSH_DIR in the FairCom documentation for a description of this parameter.					
MAXSIZE	*n* | *n*M | *n*G	O		u	w	
	Defines the maximum size of c-tree data files. Broker allocates one data file for control data and another data file for message data: *n* Maximum size in MB. *n*M Maximum size in MB. *n*G Maximum size in GB.					
PAGESIZE	*n* | *n*K	O		u	w	
	Determines how many bytes are available in each c-tree node. PSTORE COLD start is required after changing this value.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	n Same as *n*K *n*K PAGESIZE in KB. The default and minimum value is 8 KB. If PSD Reason Code = 527 is returned during UOW write processing, increase the PAGESIZE value and restart broker with PSTORE=COLD.					
PATH	A255	O		u	w	
	Path name of the target directory for c-tree index and data files.					
SYNCIO	NO | YES	O		u	w	
	Controls the open mode of the c-tree transaction log. NO c-tree transaction log is not opened in synchronous mode. Default. YES c-tree transaction log is opened in synchronous mode to improve data security. It may degrade performance of PSTORE operations, but offers the highest level of data security. See *c-tree Database as Persistent Store* in the Linux | Windows Administration documentation.					
TRACE-LEVEL	0-4	O		u	w	
	Trace level for c-tree persistent store. It overrides the global value of trace level in the attribute file. 0 No tracing. Default value. 1 Log memory allocation failures and errors during close of files. 2 n/a 3 All of trace level 1, plus UOWID in use for the various c-tree requests and function entered/exit mesages. 4 All of trace level 3, plus returned function values. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.					

SSL/TLS-specific Attributes

The Broker can use Secure Sockets Layer/Transport Layer Security (SSL/TLS) as the transport medium. The term “SSL” in this section refers to both SSL and TLS. RPC-based clients and servers, as well as ACI clients and servers, are always SSL clients. The broker is always the SSL server. For an introduction see *SSL/TLS, HTTP(S), and Certificates with EntireX*. Your operating system determines whether this section of the attribute file is required:

- **z/OS**
  The SSL-specific attribute section is not used. You can use IBM's Application Transparent Transport Layer Security (AT-TLS).
  See *Running Broker with SSL/TLS Transport* in the z/OS Administration documentation.
- **Linux and Windows**
  The SSL-specific attribute section is required, and begins with the keyword `DEFAULTS=SSL` as shown in the sample attribute file.
  The attributes in this section are needed to execute the SSL communicator of the EntireX Broker kernel.
  See also *Running Broker with SSL/TLS Transport* under Linux | Windows.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`CIPHER-SUITE`	*`string`*	O		u	w	b
	String that is passed to the underlying SSL/TLS implementation. SSL/TLS is a standardized protocol that uses different cryptographic functions (hash functions, symmetric and asymmetric encryption etc.). Some of these must be implemented in the SSL/TLS stack; others are optional. When an SSL/TLS connection is created, both parties agree by "handshake" on the cipher suite, that is, the algorithms and key lengths used. In a default scenario, this information depends on what both sides are capable of. It can be influenced by setting the attribute `CIPHER-SUITE` for the SSL/TLS server side (the broker always implements the server side). Thus stubs connect to the broker and thereby become the SSL/TLS clients. Under Linux, Windows and BS2000, the OpenSSL implementation is used. The SSL protocol is obsolete. It is no longer available. The TLS protocol is the successor of SSL and is readily available in OpenSSL. The default OpenSSL configuration uses FIPS 140-2 approved cipher suites, eligible for TLS v1.2, but without anonymous Diffie-Hellman (ADH) and pre-shared key (PSK) algorithms. The resulting set of cipher suites provides for authentication and strong encryption: `CIPHER-SUITE=FIPS+TLSv1.2:!ADH:!PSK:@STRENGTH`				

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	See *https://www.openssl.org/docs/man1.1.1/man1/ciphers*.					
CONNECTION-NONACT	*n* | *n*S | *n*M | *n*H	O		u	w	b
	Non-activity of the SSL connection, after which a close is performed and the connection resources are freed. If this parameter is not specified here, broker will close the connection only when the application (or the network itself) terminates the connection. *n* Same as *n*S. *n*S Non-activity time in seconds (min. 600, max. 2147483647). *n*M Non-activity time in minutes (min. 10, max. 35791394). *n*H Non-activity time in hours (max. 596523). If not specified, the connection non-activity is not monitored by the broker. On the stub side, non-activity can be set with the environment variable ETB_NONACT. See *Limiting the TCP/IP Connection Lifetime* under z/OS | Linux | Windows | z/VSE in the platform-specific *Administering Broker Stubs* documentation. Both sides (the stub side and the broker kernel side) act independently of each other.					
HOST	0.0.0.0 | *hostname* | *IP address*	O		u	w	b
	The address of the network interface on which broker will listen for connection requests. If HOST is not specified, broker will listen on any attached interface adapter of the system (or stack). A maximum of five HOST/PORT pairs can be specified to start multiple instances of EntireX Broker's TCP/IP transport communicator.					
KEY-FILE	*filename*	R		u	w	b
	File that contains the broker's private key (if not contained in KEY-STORE). For test purposes, EntireX delivers certificates for use on various platforms. See *SSL/TLS Sample Certificates Delivered with EntireX*. Example for Linux and Windows: MyAppKey.pem. **Note:** EntireX Broker does not support certificates of type .jks or .p12.					
KEY-PASSWD	*password* (A32)	R		u	w	b
	Password used to protect the private key. Unlocks the KEY-FILE, for example MyAppKey.pem. Deprecated. See KEY-PASSWD-ENCRYTPED below.					
KEY-PASSWD-ENCRYPTED	*encrypted value* (A64)	R		u	w	b
	Password used to protect the private key. Unlocks the KEY-FILE, for example MyAppKey.pem. This attribute replaces KEY-PASSWD to avoid a clear-text					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	password as attribute value. If `KEY-PASSWD` and `KEY-PASSWD-ENCRYTPED` are both supplied, `KEY-PASSWD-ENCRYTPED` takes precedence. Use program `etbnattr` to get the encrypted password: `etbnattr -w ssl_key_password --echo_password_only` This writes the encrypted password to standard output.					
`KEY-STORE`	*`filename`*	R		u	w	b
	SSL certificate; may contain the private key. For test purposes, EntireX delivers certificates for use on various platforms. See *SSL/TLS Sample Certificates Delivered with EntireX*. Example for Linux and Windows: *ExxAppCert.pem*. **Note:** EntireX Broker does not support certificates of type .jks or .p12.					
`MAX-MESSAGE-LENGTH`	`2147483647` | *`n`*	O		u	w	b
	Maximum message size that the broker kernel can process using transport method SSL. The default value represents the highest positive number that can be stored in a four-byte integer.					
`PORT`	`1025-65535`	O		u	w	b
	The SSL port number on which the broker will listen for connection requests. If not changed, this parameter takes the standard value as specified in the sample attribute file. If the port number is not specified, the broker will use the default value of 1958. A maximum of five `HOST/PORT` pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.					
`RESTART`	`YES` | `NO`	O		u	w	b
	`YES` The broker kernel will attempt to restart the SSL communicator (this is the default value). `NO` The broker kernel will not attempt to restart the SSL communicator.					
`RETRY-LIMIT`	`20` | *`n`* | `UNLIM`	O		u	w	b
	Maximum number of attempts to restart the SSL communicator.					
`RETRY-TIME`	`3M` | *`n`* | *`n`*`S` | *`n`*`M` | *`n`*`H`	O		u	w	b
	Wait time between suspending SSL communication due to unrecoverable error and the next attempt to restart it. *`n`* Same as *`n`*`S`. *`n`*`S` Wait time in seconds (max.2147483647).					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	*n*M Wait time in minutes (max. 35791394). *n*H Wait time in hours (max. 596523). Minimum: 1S					
REUSE-ADDRESS	YES | NO	O		u	w	b
	YES The SSL port assigned to the broker can be taken over and assigned to other applications (this is the default value). NO The SSL port assigned to the broker cannot be taken over and assigned to other applications. **Note:** This setting might be required at your site when restarting broker immediately after stopping it. This is due to the inherent latency of the TCP/IP stack when closing connections.					
STACK-NAME	*name*	O		u	w	
	Name of the TCP/IP stack that the broker is using. If not specified, broker will connect to the default TCP/IP stack running on the machine.					
TRACE-LEVEL	0-4	O		u	w	b
	The level of tracing to be performed while the broker is running with transport method SSL/TLS. It overrides the global value of trace level for all SSL/TLS routines. 0 No tracing. Default value. 1 Display IP address of incoming request, display error number of outgoing error responses. 2 All of trace level 1, plus errors if request entries could not be allocated. 3 All of trace level 2, plus all routines executed. 4 All of trace level 3, plus function arguments and return values. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.					
TRUST-STORE	*filename* | *keyring*	R		u	w	b
	Location of the store containing certificates of trust Certificate Authorities (or CAs).					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Specify the file name of the CA certificate store. Examples: `EXXCACERT.PEM`, `C:\Certs\ExxCACert.pem`					
`VERIFY-CLIENT`	`NO` | `YES`	O		u	w	b
	`YES` Additional client certificate required. `NO` No client certificate required (default). For more information see *SSL/TLS, HTTP(S), and Certificates with EntireX*.					

DIV-specific Attributes

These attributes define a persistent store that is implemented as a VSAM linear data set (LDS) accessed using Data In Virtual (DIV). This DIV persistent store is a container for units of work. The DIV-specific attribute section begins with the keyword `DEFAULTS = DIV`. The attributes in this section are required if `PSTORE-TYPE = DIV` is specified.

Note: All attributes except the deprecated `DIV` were introduced with EntireX version 9.12. They replace the *Format Parameters* of earlier versions, which are deprecated but still supported for compatibility reasons.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`DIV`	`A511`	O	z			
	The VSAM persistent store parameters, enclosed in double quotes ("""). The value can span more than one line. **Note:** Deprecated. This attribute is applicable only if you are supplying the persistent store parameters using *Format Parameters* of earlier versions. We recommend you use the attributes below that were introduced with EntireX 9.12 instead.					
`DATASPACE-NAME`	`A8`	O	z			
	Defines the name of the dataspace that will be used to map the persistent store. Default value is `DSPSTORE`.					
`DATASPACE-PAGES`	`126-524284`	O	z			
	Defines the size of the dataspace used to map the persistent store (size=`DATASPACE-PAGES` * 4 KB). We recommend using the maximum value. Default value is 2048.					
`DDNAME`	`A8`	R	z			
	Defines the JCL `DDNAME` that will be used to access the persistent store.					
`STORE`	`A8`	R	z			
	Defines an internal name that is used to identify the persistent store.					
`TRACE-LEVEL`	`0-4`	O	z			
	Trace level for DIV. It overrides the global value of trace level in the attribute file. `0` No tracing. Default value. `1` Log selected `DIV SAVE` calls taking longer than 2 seconds elapsed time. `2` n/a `3` All of trace level 1, plus UOWID in use for the various DIV requests. `4` n/a					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the `TRACE-LEVEL` attribute, you must restart the broker for the change to take effect. For temporary changes to `TRACE-LEVEL` without a broker restart, use the EntireX Broker command-line utility `ETBCMD`.					

Adabas-specific Attributes

The Adabas-specific attribute section begins with the keyword `DEFAULTS = ADABAS`. The attributes in this section are required if `PSTORE-TYPE = ADABAS` is specified. In previous versions of EntireX, these Adabas-specific attributes and values were specified in the broker-specific `PSTORE-TYPE` attribute.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`BLKSIZE`	`126-20000`	O	z	u	w	b
	Optional blocking factor used for message data. If not specified, broker will split the message data into 2 KB blocks to be stored in Adabas records. The maximum value depends on the physical device assigned to data storage. See the *Adabas* documentation. For reasons of efficiency, do not specify a `BLKSIZE` much larger than the actual total size of the UOW data to be written. The total UOW size is the sum of all messages in the UOW plus 41 bytes of header information. This takes effect only after COLD start. The `BLKSIZE` parameter applies only for a cold start of broker; subsequently the value of `BLKSIZE` is taken from the last cold start. Default value is 2000.					
`DBID`	`1-32535`	R	z	u	w	b
	Database ID of Adabas database where the persistent store resides.					
`FNR`	`1-32535`	R	z	u	w	b
	File number of broker persistent store file.					
`FORCE-COLD`	`N` | `Y`	O	z	u	w	b
	Determines whether a broker cold start is permitted to overwrite a persistent store file that has been used by another broker ID and/or platform. Specify `Y` to allow existing information to be overwritten.					
`MAXSCAN`	`0` | *n*	O	z	u	w	b
	Limits display of persistent UOW information in the persistent store through Command and Information Services. Default value is 1000.					
`OPENRQ`	`N` | `Y`	O	z	u	w	b
	Determines whether driver for Adabas persistent store is to issue an `OPEN` command to Adabas.					
`SVC`	`200-255`	R	z			
	Use this parameter to specify the Adabas SVC number to be used by the Adabas persistent store driver.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
TRACE-LEVEL	0-4	O	z	u	w	b
	Trace level for Adabas persistent store. It overrides the global value of trace level in the attribute file. 0 No tracing. Default value. 1 Log selected Adabas CB fields (command code, response code, subcode, ISN, additions). 2 n/a 3 All of trace level 1, plus UOWID in use for the various Adabas requests and function entered/exit mesages. 4 All of trace level 3, plus more Adabas CB fields for successful requests and returned function values. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.					

Application Monitoring-specific Attributes

The application monitoring-specific attribute section begins with the keyword `DEFAULTS=APPLICATION-MONITORING`. It contains attributes that apply to the application monitoring functionality. At startup time, the attributes are read if the Broker-specific attribute `APPLICATION-MONITORING=YES` is specified. Duplicate or missing values are treated as errors. When an error occurs, application monitoring is turned off and EntireX Broker continues execution. See the separate Application Monitoring documentation.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`APPLICATION-MONITORING-NAME` or `APPMON-NAME`	`A100`	O	z	u	w	b
	Specifies a default application monitoring name. Used to set the value of the **ApplicationName** KPI.					
`COLLECTOR-BROKER-ID`	`A64`	R	z	u	w	b
	Identifies the Application Monitoring Data Collector. Has the format *`host_name`*:*`port_number`*, where where *`host_name`* is the host where the Application Monitoring Data Collector is running, and *`port_number`* is the port number of the Application Monitoring Data Collector. The default port is 57900.					
`TRACE-LEVEL`	`0-4`	O	z	u	w	b
	The level of tracing to be performed while the broker is running with application monitoring. `0` No tracing. Default value. `1` Display application monitoring errors. `2` All of trace level 1, plus measuring points for application monitoring. `3` All of trace level 2, plus function entered/exit messages with argument values and monitoring buffers. `4` All of trace level 3, plus returned function values. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the `TRACE-LEVEL` attribute, you must restart the broker for the change to take effect. `TRACE-LEVEL` cannot be changed dynamically for application monitoring.					

Authorization Rule-specific Attributes

The authorization rule-specific attribute section begins with the keyword `DEFAULTS=AUTHORIZATION-RULES`. It contains attributes that enhance security-related definitions. At startup time, the attributes are read if the following conditions are met:

- Broker-specific attribute `SECURITY=YES`
- Security-specific attributes `SECURITY-SYSTEM=OS` and `SECURITY-LEVEL=AUTHORIZATION`

When an error occurs, the EntireX Broker stops. See *Authorization Rules*.

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`RULE-NAME`	`A32`	R		u	w	
	Specifies a rule name. A rule is a container for a list of services and a list of client and server user IDs. All users defined in a rule are authorized to use all services defined in this rule. See example under *Rules Stored in Broker Attribute File*.					

`CLASS`
`SERVER`
`SERVICE`
	`A32`	R		u	w	

	These three attributes together identify the service. `CLASS` must be specified first, followed immediately by `SERVER` and `SERVICE`. ***Wildcard Service Definitions*** are allowed.					
`CLIENT-USER-ID`	`A32`	R		u	w	
	Defines an authorized client user ID.					
`SERVER-USER-ID`	`A32`	R		u	w	
	Defines an authorized server user ID.					

Variable Definition File

The broker attribute file contains the configuration of one EntireX Broker instance. In order to share attribute files between different brokers, you identify the attributes that are unique and move them to a variable definition file. This file enables you to share one attribute file among different brokers. Each broker in such a scenario requires its own variable definition file.

The following attributes are considered unique for each machine:

- `BROKER-ID` (in ***Broker-specific Attributes***)
- `NODE` (in ***Adabas SVC/Entire Net-Work-specific Attributes***)
- `PORT` (in ***SSL/TLS-specific Attributes*** and ***TCP/IP-specific Attributes***)

How you use the variable definition file will depend upon your particular needs. For instance, some optional attributes may require uniqueness - for example, `DBID` and `FNR` in `DEFAULTS=ADABAS` - so that you may specify the persistent store.

III Broker Command and Information Services

7 Broker Command and Information Services

- CIS Overview Table 106
- Modes of Requesting the Services 107
- ETBCMD: Executable Command Requests 109
- ETBINFO: Returnable Information Requests 114

EntireX Broker provides two internal services: Command Service and Information Services that can be used administer and monitor the EntireX Broker. The command service allows you to issue a set of Broker commands; the information services provide you with various statistics to better administer and tune your Broker. Because these services are implemented internally, nothing has to be started or configured. You can use these services immediately after starting EntireX Broker.

See also *Broker CIS Data Structures* in the ACI Programming documentation.

CIS Overview Table

EntireX Broker provides these predefined internal services:

- **Command Service**
  Provides a facility to issue commands against the Broker (e.g. `SHUTDOWN` etc.).
- **Information Services**
  Provides a query mechanism to obtain various types of information on the Broker, which is helpful for administration and tuning.

Since these services are implemented internally, nothing has to be started, configured or defined in the Broker attribute file. You can use them immediately after starting the Broker. They can be requested as follows:

Mode of Request	Tools	Services	Requirements
User-Written Interface	application program	■ `INFO` ■ `USER-INFO` ■ `CMD` ■ `PARTICIPANT-SHUTDOWN` ■ `SECURITY-CMD`	■ request structures
Command-line Utilities	`ETBINFO` utility	■ `INFO` ■ `USER-INFO`	■ profile ■ command-line parameters
	`ETBCMD` utility	■ `CMD` ■ `PARTICIPANT-SHUTDOWN` ■ `SECURITY-CMD`	■ command-line parameters

Applicable operating systems: z/OS, Linux and Windows.

Description of Services

INFOand USER-INFO

- `INFO` is the full information service. Specify it for the full information service. All clients, servers and conversations are listed.
- `USER-INFO` is limited to your user-specific information. Specify it for limited information service. Only the user's own resources are listed.

CMD, PARTICIPANT-SHUTDOWNand SECURITY

- `CMD` is the full command service.
- `PARTICIPANT-SHUTDOWN` is limited to shutting down participants.
- `SECURITY-CMD` is limited to EntireX Security-related commands.

Modes of Requesting the Services

Use one of these three modes to request a service:

- Command-line Utilities
- Graphical User Interface
- User-Written Interface

The method for requesting these services is the same as the method for requesting any other service. For both types of services, an application issues a `SEND` command with appropriate data and retrieves a reply. The request itself is specified within the `SEND` buffer; the reply - if there is one - is specified in the `RECEIVE` buffer.

For Information Services requests, `RECEIVE` operations must be repeated until the Information Service indicates the end of data with an `EOC` return message.

Command-line Utilities

Software AG provides three command-line utility programs for use with EntireX Broker. All utility programs use command-line parameters that specify various options and information to be built into a request. These utility programs are:

- `ETBINFO`
  Queries the Broker for different types of information, generating an output text string with basic formatting. This text output can be further processed by script languages (or elsewhere). `ETBINFO` uses data descriptions called profiles to control the type of data that is returned for a request. `ETBINFO` is useful for configuring and administering EntireX Broker efficiently - e.g., how many

users are to run concurrently and whether the number of specified message containers is large enough.

See `ETBINFO` under *Broker Command-line Utilities* in the platform-specific Administration documentation for profiles, examples and utility parameters.

- `ETBCMD`
  Allows you to take actions - e.g., purge a unit of work, stop a server, shut down a Broker - against EntireX Broker.

  See `ETBCMD` under *Broker Command-line Utilities* in the platform-specific Administration documentation for utility parameters.

Version Information

- The `ETBINFO` and `ETBCMD` CIS command-line utilities are compatible with all versions of EntireX Broker.
- Display keywords applying to a specific version of Broker will not be returned when a call is made to any older version of Broker.

Graphical User Interface

Software AG provides a graphical user interface, Command Central, for displaying information on the Broker and/or executing administrative functions. Software AG Command Central is a tool that enables you to manage your Software AG products remotely from one location. Command Central offers a browser-based user interface, but you can also automate tasks by using commands to remotely execute actions from a terminal or custom script (for example CI servers such as Jenkins, or generic configuration management tools such as Puppet or Chef).

Command Central can assist with the following configuration, management, and monitoring tasks:

- Infrastructure engineers can see at a glance which products and fixes are installed, where they are installed, and compare installations to find discrepancies.
- System administrators can configure environments by using a single web user interface or command-line tool. Maintenance involves minimum effort and risk.
- Release managers can prepare and deploy changes to multiple servers using command-line scripting for simpler, safer lifecycle management.
- Operators can monitor server status and health, as well as start and stop servers from a single location. They can also configure alerts to be sent to them in case of unplanned outages.

User-Written Interface

If you access the Command and Information Services through a user-written application, you must use a defined protocol. This protocol describes the structures needed to communicate with the service(s) so that the request is correctly interpreted by the Broker.

See *Writing Applications: Command and Information Services* and *Broker CIS Data Structures*.

ETBCMD: Executable Command Requests

The following command requests can be issued, using `ETBCMD`. All the functions listed in this table are applicable to all three request modes; see ***Modes of Requesting the Services***.

Note: Version numbers in this table refer to the interface version and not to the Broker version.

Command Request	Comment	CIS Interface Version
`APPMON-OFF`	Turn off the Application Monitoring feature in Broker. In addition to changing the current status, `APPLICATION-MONITORING=NO` is written to the Broker attribute file.	11
`APPMON-ON`	Turn on the Application Monitoring feature in Broker. You must specify the collector broker ID. In addition to changing the current status, `APPLICATION-MONITORING=YES` is written to the Broker attribute file.	11
`ALLOW-NEWUOWMSGS`	New UOW messages are allowed.	3
`CLEAR-CMDLOG-FILTER`	Remove the specified command log filter.	5
`CONNECT-PSTORE`	Connects the persistent store. See *Availability of Persistent Store*.	4
`DISABLE-ACCOUNTING`	Disables accounting. Accounting records are discarded until accounting is enabled.	5
`DISABLE-CMDLOG`	Disable command logging.	5
`DISABLE-DYN-WORKER`	Disable the `DYNAMIC-WORKER-MANAGEMENT`. `DYNAMIC-WORKER-MANAGEMENT=YES` must be configured in the attribute file. The current number of active worker tasks will not be changed until	7

Command Request	Comment	CIS Interface Version
	DYNAMIC-WORKER-MANAGEMENT is enabled again.	
DISCONNECT-PSTORE	Disconnects the persistent store. See *Availability of Persistent Store*.	4
ENABLE-ACCOUNTING	Enable accounting.	5
ENABLE-CMDLOG	Enable command logging.	5
ENABLE-DYN-WORKER	Enable the DYNAMIC-WORKER-MANAGEMENT again. DYNAMIC-WORKER-MANAGEMENT=YES must be configured in the attribute file. DYNAMIC-WORKER-MANAGEMENT has been disabled before. Additional worker tasks can be started again, or stopped if not used.	
FORBID-NEWUOWMSGS	New UOW messages are not allowed.	3
PRODUCE-STATISTICS	Output current statistics to the broker log.	5
PURGE	Remove a unit of work from the persistent store.	2
RESET-USER	Clear all cached security information for the specified user ID.	5
REFRESH-RULES	Read section AUTHORIZATION-RULES from broker attribute file and replace active rules with these new rules.	12
RESUME	Transport ID: NET|S*nn*|T*nn*. Resume a suspended transport communicator. If the communicator was not suspended before, an error message will be returned.	
SET-CMDLOG-FILTER	Add the specified command log filter.	5
SET-COLLECTOR	Set the collector broker ID in Broker. COLLECTOR-BROKER-ID=*value* is written to the Broker attribute file. If the APPLICATION-MONITORING section is not already defined in the attribute file, the section is added, that is, a line containing DEFAULTS = APPLICATION-MONITORING followed by attribute COLLECTOR-BROKER-ID=*value*.	11

Command Request			Comment	CIS Interface Version
SET-UOW-STATUS		PSF	Set the status of postponed UOWs to ACCEPTED or CANCELLED, for example: etbcmd -b<broker_id> ↩ -cSET-UOW-STATUS -dPSF -oACCEPTED ↩ -n<class>/<server>/<service>	10
SHUTDOWN	BROKER		Shutdown Broker immediately.	1
	CONVERSATION <conversation-id>	Command applies to conversations without units of work only. The security rights shutting down the service are required for shutting down the conversation.		7
		IMMED	The specified conversation is immediately removed. All messages of the conversation are lost.	
		QUIESCE	An end of conversation is issued. The conversation remains active.	
	SERVER	IMMED	Shutdown server immediately. The server must be uniquely identified using field P-USER-ID or SEQNO and will be completely removed from the broker environment. The following steps will be performed: ■ Error code 00100050 will be returned to the server, if it is waiting. ■ All existing conversations will be finished with EOC. ■ User will be logged off.	1
		QUIESCE	Shutdown server but allow existing conversations to continue. The termination is signaled to the server by error code 00100051. After this, the next call issued must be a DEREGISTER for all services (SC=*, SN=*, SV=* if more than one service is active).	
	SERVICE <class/server/service>	Internal services cannot be shut down.		7
		IMMED	Caution: All servers offering this service will be deregistered and logged off. The following steps will be performed: ■ Error code 00100050 will be replied to all servers, if they are waiting.	

Command Request			Comment	CIS Interface Version
			■ All existing conversations will be finished with EOC. ■ Users will be logged off.	
		QUIESCE	All servers offering this service are deregistered. Shutdown servers but allow existing conversations to continue. The termination is signaled to the servers by error code 00100051. After this, the next call issued must be a DEREGISTER for the service.	
	PARTICIPANT	IMMED	Shutdown participant immediately. The participant must be identified, using fields P-USER-ID, UID TOKEN or SEQNO and will be completely removed from the Broker environment. See *Broker CIS Data Structures*. The following steps will be performed: ■ Error code 00100050 will be replied to the participant, if it is waiting. ■ All existing conversations will be finished with EOC. ■ User will be logged off. Within EntireX Broker nomenclature, a participant is an application implicitly or explicitly logged on to the Broker as a specific user. See *Implicit Logon* and *Explicit Logon*. A participant could act as client or server.	4
		QUIESCE	Shutdown participant but allow existing conversations to continue. The termination is signaled to the participant by error code 00100051.	
START	TRANSPORT	Transport ID: NET|S*nn*|T*nn*	Start a transport communicator that was previously stopped. If the communicator was not stopped before, an error message will be returned.	7
STATUS	TRANSPORT	Transport ID: NET|S*nn*|T*nn*	Check the current status of the transport communicator.	7
STOP	TRANSPORT	Transport ID: NET|S*nn*|T*nn*	Stop an active or suspended transport communicator. The transport communicator	7

Command Request			Comment	CIS Interface Version
			will shut down. All transport-specific resources will be freed. User requests receive response code 148.	
SUSPEND	TRANSPORT	Transport ID: NET|S*nn*|T*nn*	Suspend an active transport communicator.	7
SWITCH-CMDLOG			Force a switch of command logging output files.	5
TRACE-FLUSH	BROKER		Flush all trace data kept in internal trace buffers to stderr (DD:SYSOUT). The broker-specific attribute TRMODE=WRAP is required.	7
TRACE-OFF	BROKER		Set TRACE-LEVEL off in Broker.	1
	PSF		Set TRACE-LEVEL off in persistent store.	5
	SECURITY		Set TRACE-LEVEL off in EntireX Security.	5
TRACE-ON	BROKER		Set TRACE-LEVEL on in Broker. Values: 1 | 2 | 3 | 4.	1
	PSF		Set TRACE-LEVEL on in persistent store. Values: 1 | 2 | 3 | 4.	5
	SECURITY		Set TRACE-LEVEL on in EntireX Security. Values: 1 | 2 | 3 | 4.	5
TRAP-ERROR	BROKER	Error number: *nnnn*	Modifies the setting of the broker-specific attribute TRAP-ERROR.	7

ETBINFO: Returnable Information Requests

The following information requests can be returned. All the functions listed in this table are applicable to all three request modes (see *Modes of Requesting the Services*). The returned data is described under *Information Reply Structures* in the ACI Programming documentation.

Note: Version numbers in this table refer to the interface version and not to the Broker version.

Information Request	Comment	Interface Version
BROKER	Global information on this Broker. No additional selection criteria are needed. Other selection criteria fields are ignored.	1
CLIENT	Information on active clients.	1
CMDLOG-FILTER	Information on command log filters.	5
CONVERSATION	Information on active conversations.	1
NET	Information on the Entire Net-Work communicator.	5
POOL	Information on Broker pool usage and dynamic memory management.	7
PSF	Information on a unit of work's status and Information for persistent store.	2
PSFDIV	Global information on the DIV persistent store.	2
PSFADA	Global information on the Adabas persistent store.	3
PSFCTREE	Global information on the c-tree persistent store.	5
RESOURCE	Information on Broker resource usage.	7
SECURITY	Global information on EntireX Security.	5
SERVER	Information on active servers.	1
SERVICE	Information on active services.	1
SSL	Information on the SSL communicator.	5
STATISTICS	Statistics on selected Broker resources.	7
TCP	Information on the TCP/IP communicator.	5
UOW-STATISTICS	Statistics on UOWs of selected services.	9
USER	Information on all users of Broker regardless of the user type.	7
WORKER	Global information on all workers. No additional selection criteria are needed. Other selection criteria fields are ignored.	1
WORKER_USAGE	Information on usage of worker tasks and dynamic worker management.	7

IV EntireX Broker Reporting

This chapter details the reporting options of EntireX Broker.

Configuration Report

EntireX Broker reads configuration information from an attribute file during startup. In order to reduce the number of different attribute files, you may define a global attribute file and specify the individual settings within a variable definitions file. Thus unique attributes like `BROKER-ID` and `PORT` are kept as part of the variable definitions file, while other parameters such as service definitions can be shared among a group of Broker instances. This feature is described in detail in *Variable Definition File*.

In the past there was a one-to-one relationship between Brokers and attribute files. To determine your Broker configuration, you could reference your attribute file. Now that you may create a global attribute file and substitute parameters at startup, it may not be clear what configuration was used to start your Broker. To see the exact configuration used at startup, you can now view the configuration report for each Broker. The configuration report will display exactly which values were used for each attribute at startup.

Here is a sample configuration report:

```
EntireX 8.0.0.12     Configuration Report      2007-10-02 08:56:23    Page     1

Variable definitions file:
   1: BID = ETB191
   2: N   = 113
   3: P   = HOT
   4: PCA = localhost:3938:SSL
   5: PT  = ADABAS
   6: RM  = STANDARD
   7: SP  = 3939
   8: TP  = 3930
   9: TR  = SSL-TCP-NET
```

```
EntireX 8.0.0.0     Configuration Report      2007-10-02 08:56:23    Page     2

Attribute file:
   1: ***********************************************************************
   2: *                                                                     *
   3: *                    EntireX Broker Attribute File                    *
   4: *                                                                     *
   5: ***********************************************************************
   6:
   7: ******************** Global section ************************************
   8:
   9: DEFAULTS = BROKER
  10:   ABEND-MEMORY-DUMP                = NO
  11:   ACCOUNTING                       = NO
  12:   AUTOLOGON                        = YES
  13:   BROKER-ID                        = ${BID}
- - - - Substitution: ${BID} = ETB191
  14:   CLIENT-NONACT                    = 15M
```

The contents of the variable definitions file and the contents of the attribute file are copied to this configuration report. In addition, all variables in the attribute file will be appended by another line reporting the effective value for the variable. Thus, it's possible to keep track of the substitution procedure.

On Linux and Windows, a file called `CONFIG.REPORT` is created in the current working directory of Broker. The environment variable `ETB_CONFIG_REPORT` may contain an alternative location. However, on z/OS, `DDNAME ETBCREP` is required to assign an output file for this report. Any failure will trigger a diagnostic message in the Broker log.

Load Module Report

The Load Module Report is created during startup of EntireX Broker on z/OS. All modules in all data sets concatenated to the `STEPLIB` chain for Broker execution are listed.

```
Operating System: z/OS 06.00
Node Name:        DAEF
IPL Date:         2007-10-02
IPL Time:         07:19:21
CPU Model:        2096

EntireX 8.0.0.12     Load Module Report 2007-10-02 08:56:23    Page     1
Total  Module   Date       Time  VRSPP Build number     Alias Level CurNo
                Steplib level 0: SAG.EXB731.LOAD
    1  ADAACK                                            NO      0     1
    2  ADABSP                                            NO      0     2
    3  ADACDC                                            NO      0     3
    4  ADACLU                                            NO      0     4
    5  ADACLX                                            NO      0     5
```

```
    6  ADACMO                                          NO      0     6
    7  ADACMP                                          NO      0     7
    8  ADACMR                                          NO      0     8
    9  ADACMU                                          NO      0     9
   10  ADACNS                                          NO      0    10
   11  ADACNV                                          NO      0    11
...
  156  ETBCMD   2007-10-01 15.48 73012 2007-10-01 15:01   NO      0   156
  157  ETBINFO  2007-10-01 15.48 73012 2007-10-01 15:01   NO      0   157
  158  ETBMISC                                          NO      0   158
  159  ETBNATTR 2007-10-01 15.48 73012 2007-10-01 15:01   NO      0   159
  160  ETBNUC   2007-10-01 15.48 73012 2007-10-01 15:01   NO      0   160
```

This report provides STEPLIB level, date, and time stamps if a certain pattern is used for the module structure. `DDNAME ETBMREP` must be assigned to get this report.

Storage Report

You can create an optional report file that provides details about all activities to allocate or to deallocate memory pools. This section details how to create the report and provides a sample report.

- Creating a Storage Report
- Platform-specific Rules
- Sample Storage Report

See also Broker-specific attribute `STORAGE-REPORT`.

Creating a Storage Report

Use Broker's global attribute `STORAGE-REPORT` with the value `YES`. If attribute value `YES` is supplied, all memory pool operations will be reported if the output mechanism is available. If the value `NO` is specified, no report will be created.

Platform-specific Rules

z/OS

`DDNAME ETBSREP` assigns the report file. Format `RECFM=FB, LRECL=121` is used.

Linux and Windows

Broker creates a file with the name *STORAGE.REPORT* in the current working directory. If the environment variable `ETB_STORAGE_REPORT` is supplied, the file name specified in the environment variable will be used. If Broker receives the command-line argument `-r`, the token following argument `-r` will be used as the file name.

BS2000

`LINK-NAME ETBSREP` assigns the report file. Format `REC-FORM=V, REC-SIZE=0, FILE-TYPE ISAM` is used by default.

Sample Storage Report

The following is an excerpt from a sample STORAGE report.

```
EntireX 8.1.0.00     STORAGE Report     2009-06-26 12:28:58    Page     1

Identifier     Address        Size         Total         Date      Time    Action
KERNEL POOL  0x25E48010   407184 bytes   407184 bytes  2009-06-26 12:...  Allocated
HEAP POOL    0x25EB4010  1050692 bytes  1457876 bytes  2009-06-26 12:...  Allocated
...
```

Header	Description
Identifier	Name of the memory pool.
Address	Start address of the memory pool.
Size	Size of the memory pool.
Total	Total size of all obtained memory pools.
Date, Time	Date and time of the action.
Action	The action of Broker. The following actions are currently supported: Allocated: memory pool is allocated. Deallocated: memory pool is deallocated.

Persistent Store Report

You can create an optional report file that provides details about all records added to or deleted from the persistent store. This section details how to create the report and provides a sample report.

- Configuration

- Sample Report

Configuration

To create a persistent store report, use Broker's global attribute `PSTORE-REPORT` with the value `YES`.

When the attribute value `YES` is supplied, all created or deleted persistent records will be reported if the output mechanism is available.

If the value `NO` is specified, no report will be created.

The report file is created using the following rules:

BS2000

`LINK-NAME ETBPREP` assigns the report file. Format `REC-FORM=V, REC-SIZE=0, FILE-TYPE ISAM` is used by default.

Linux

Broker creates a file with the name *PSTORE.REPORT* in the current working directory.

If the environment variable `ETB_PSTORE_REPORT` is supplied, the file name specified in the environment variable will be used.

If Broker receives the command-line argument `-p`, the token following argument `-p` will be used as the file name.

Windows

Same as Linux.

z/OS

`DDNAME ETBPREP` assigns the report file. Format `RECFM=FB, LRECL=121` is used.

Sample Report

The following is an excerpt from a sample `PSTORE` report.

```
EntireX 10.9     PSTORE Report      2016-10-18 10:46:18    Page     1

Identifier        Elements  Total length  Record Type   Date          Action
0000000000000000        1           760   Master        2016-10-18... Created
0010000000000001        1          5022   Conversation  2016-10-18... Created
0010000000000002        1          5022   Conversation  2016-10-18... Created
0010000000000003        1          5022   Conversation  2016-10-18... Created
0010000000000001                          Conversation  2016-10-18... Postponed
0010000000000001                          Conversation  2016-10-18... Accepted
```

```
0010000000000002                          Conversation  2016-10-18... Postponed
0010000000000002                          Conversation  2016-10-18... Accepted
0010000000000003                          Conversation  2016-10-18... Postponed
0010000000000003                          Conversation  2016-10-18... Accepted
0010000000000003                          Conversation  2016-10-18... Postponed
0010000000000003                          Conversation  2016-10-18... Accepted
0010000000000001                          Conversation  2016-10-18... Deleted
0010000000000002                          Conversation  2016-10-18... Deleted
0010000000000003                          Conversation  2016-10-18... Deleted
```

The following fields are provided in the report:

- `Identifier` provides the UOWID (record ID).
- `Elements` gives the number of messages per UOW when creating or loading records.
- `Total Length` gives the size of the raw record when creating or loading records.
- `Record Type` describes the type of the data. Following types are currently supported:
  - `Cluster`: a special record for synchronization purposes
  - `Conversation`: a unit of work as part of a conversation
  - `Master`: a special record to manage the persistent store
- Date and time of the action
- `Action` describes the action of Broker. The following actions are currently supported:
  - `Accepted`: UOW status was changed from `POSTPONED` to `ACCEPTED`
  - `Created`: record is created
  - `Deleted`: record is deleted
  - `Postponed`: UOW status was changed from `DELIVERED` to `POSTPONED`
- Remaining postpone attempts.

License Report

The License Report is created during broker startup on the respective platform. It contains the contents of the license file itself and some machine data.

z/OS

DDNAME `ETBLREP` must be assigned to get this report. See *Step 2: Edit the Broker Startup Procedure*.

BS2000

`LINK-NAME ETBLREP` must be assigned to get this report.

8 Command Logging in EntireX

- Introduction to Command Logging 122
- Command Log Filtering using Command-line Interface ETBCMD 125
- ACI-driven Command Logging 127
- Dual Command Log Files 127

Command logging is a feature to assist in debugging Broker ACI applications. A command in this context represents one user request sent to the Broker and the related response of Broker.

Command logging is a feature that writes the user requests and responses to file in a way it is already known with Broker trace and `TRACE-LEVEL=1`. But command logging works completely independent from trace, and data is written to a file only if defined command trace filters detect a match.

Broker stub applications send commands or requests to the Broker kernel, and the Broker kernel returns a response to the requesting application. Developers who need to resolve problems in an application need access to those request and response strings inside the Broker kernel. That's where command logging comes in. With command logging, request and response strings from or to an application are written to a file that is separate from the Broker trace file. Command logging works based on defined filters. Nothing is logged if there are no filters. If filters are defined and if there is a match, this user request is logged.

Note: All applied filters are lost after Broker restart and have to be applied again.

Introduction to Command Logging

This section provides an introduction to command logging in EntireX and offers examples of how command logging is implemented. It covers the following topics:

- Overview
- Command Log Files
- Defining Filters
- Programmatically Turning on Command Logging

Overview

Command logging is similar to a Broker trace that is generated when the Broker attribute `TRACE-LEVEL` is set to 1. Broker trace and command logging are independent of each other, and therefore the configuration of command logging is separate from Broker tracing.

The following Broker attributes are involved in command logging:

Attribute	Description
`CMDLOG`	Set this to "N" if command logging is not needed.
`CMDLOG-FILE-SIZE`	A numeric value indicating the maximum size of command log file in KB.
`NUM-CMDLOG-FILTER`	The maximum number of filters that can be set.

In addition to `CMDLOG=YES`, the Broker needs the assignment of the dual command logging files during startup. If these assignments are missing, Broker will set `CMDLOG=NO`. See also *Broker Attributes*.

Command Log Files

The Broker keeps a record of commands (request and response strings) in a command log file.

At Broker startup, you will need to supply two command log file names and paths. Only one file is open at a time, however, and the Broker writes commands (requests and responses) to this file.

Under Linux and Windows, the startup options `-y` and `-z` are evaluated by executable `etbnuc`. These options are used to specify the command log file names. Startup script/service assign these files by default.

Under z/OS, the file requirements are two equally sized, physical sequential files defined with a record length of 121 bytes, i.e. `DCB=(LRECL=121,RECFM=FB,BLKSIZE=nnnn)`. We recommend you allocate files with a single (primary) extent only. For example `SPACE=(CYL,(30,0))`. The minimum file size is approximately 3 cylinders of 3390 device. Alternatively, the dual command log files can be allowed in USS HFS file system. The following points apply to z/OS only:

- If the value of `CMDLOG-FILE-SIZE` is lower than the capacity of the CMDLOG data set, `CMDLOG-FILE-SIZE` is used.
- If the value of `CMDLOG-FILE-SIZE` is greater than the capacity of the CMDLOG data set, `CMDLOG-FILE-SIZE` is adjusted to the capacity of the data set.
- If no `CMDLOG-FILE-SIZE` was defined, it is set to the capacity of the CMDLOG data set.

When the size of the active command log file reaches the KB limit set by `CMDLOG-FILE-SIZE`, the file is closed and the second file is opened and becomes active. When the second file also reaches the KB limit set by `CMDLOG-FILE-SIZE`, the first file is opened and second file is closed. Existing log data in a newly opened file will be lost.

Defining Filters

In command logging, a filter is used to store and identify a class, server, or service, as well as a user ID.

Use the command-line tool `etbcmd` to define a filter. During processing, the Broker evaluates the class, server, service, and user ID associated with each incoming request and compares them with the same parameters specified in the filters. If there is a match, the request string and response string of the request is printed out to the command log file.

Programmatically Turning on Command Logging

Applications using ACI version 9 or above have access to the new field `LOG-COMMAND` in the ACI control block.

If this field is set, the accompanying request and the Broker's response to this request is logged to the command log file.

Note: Programmatic command logging ignores any filters set in the kernel.

Command Log Filtering using Command-line Interface ETBCMD

The examples assume that Broker has been started with the attribute `CMDLOG=Y`.

- Setting Filters
- Deleting Filters
- Disabling and Enabling a Filter

Setting Filters

Filters need to be set before running the stub applications whose commands are to be logged. Filter for class, server, service may contain fully qualified names (`ACLASS/ASERVER/ASERVICE`) or asterisk for any (e.g. `ACLASS/*/ASERVICE`). Partially qualified filter names (`ACLA*/ASERVER/ASERV*`) are not supported.

Linux and Windows

Command	Description
`etbcmd -blocalhost:1970:TCP -cSET-CMDLOG-FILTER -dBROKER -xuser -nACLASS/ASERVER/ASERVICE`	This command sets filters on `ACLASS/ASERVER/ASERVICE`. All ACI calls issued by *all* users to this service will be logged.
`etbcmd -blocalhost:1970:TCP -cSET-CMDLOG-FILTER -dBROKER -xuser -nACLASS/ASERVER/ASERVICE -Usaguser1`	This command set filters on `ACLASS/ASERVER/ASERVICE` and user ID `saguser1`. All ACI calls to this service *as well as* those issued by `saguser1` will be logged.

z/OS

Command	Description

`//ETBCMD EXEC PGM=ETBCMD,`
`// PARM=('/-blocalhost:1970:TCP ↩`
`-cSET-CMDLOG-FILTER -xuser ',`
`//  '-dBROKER ↩`
`-nACLASS/ASERVER/ASERVICE')`
	This command sets filters on `ACLASS/ASERVER/ASERVICE`. All ACI calls issued by *all* users to this service will be logged.

`//ETBCMD EXEC PGM=ETBCMD,`
`// PARM=('/-blocalhost:1970:TCP ↩`
`-cSET-CMDLOG-FILTER -xuser ',`
`//  '-dBROKER -nACLASS/ASERVER/ASERVICE ↩`
`-Usaguser1')`
	This command sets filters on `ACLASS/ASERVER/ASERVICE` and user ID `saguser1`. All ACI calls to this service *as well as* those issued by `saguser1` will be logged.

Note: If more than one service is set as a filter, all ACI calls sent to any of these services will be logged. Identical filters cannot be set. Attempts to set a second filter that matches an existing filter will be rejected. Similarly, the maximum number of filters that can be added is

defined in NUM-CMDLOG-FILTER. If the maximum number of filters is already being used, delete an existing filter to make room for a new filter.

Deleting Filters

The following provides an example of how to delete an existing filter on a service.

To delete a filter

- Enter the following command.

  Under Linux:

```
etbcmd -d BROKER -b localhost:1970:TCP -c CLEAR-CMDLOG-FILTER ↩
-nACLASS/ASERVER/ASERVICE -U saguser1
```

  Under z/OS:

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-blocalhost:1970:TCP -cCLEAR-CMDLOG-FILTER -xuser ',
//       '-dBROKER -nACLASS/ASERVER/ASERVICE')
```

  If the filter does not exist, the command will return an error.

Disabling and Enabling a Filter

Filters can be set and still be disabled (made inactive).

To disable a filter

- Enter the following command.

  Under Linux:

```
etbcmd -blocalhost:1970:TCP -cDISABLE-CMDLOG-FILTER -dBROKER -xuser ↩
-nACLASS/ASERVER/ASERVICE -Usaguser1
```

  Under z/OS:

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-blocalhost:1970:TCP -cDISABLE-CMDLOG-FILTER -xuser ',
//       '-dBROKER -nACLASS/ASERVER/ASERVICE -Usaguser1')
```

Note: A disabled filter will not bring down the count of filters in use.

To enable a filter

- Enter the following command to enable the disabled filter.

  Under Linux:

```
etbcmd -blocalhost:1970:TCP -cENABLE-CMDLOG-FILTER -dBROKER -xuser ↩
-nACLASS/ASERVER/ASERVICE -Usaguser1
```

  Under z/OS:

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-blocalhost:1970:TCP -cENABLE-CMDLOG-FILTER -xuser ',
//       '-dBROKER -nACLASS/ASERVER/ASERVICE -Usaguser1')
```

ACI-driven Command Logging

EntireX components that communicate with Broker can trigger command logging by setting the field `LOG-COMMAND` in the ACI control block.

When handling ACI functions with command log turned on, Broker will not evaluate any filters. Application developers must remember to reset the `LOG-COMMAND` field if subsequent requests are not required to be logged.

Dual Command Log Files

Broker's use of two command log files prevents any one command log file from becoming too large. What you need to specify depends on the operating system:

z/OS

When starting a Broker with command log support, you must therefore specify two data sets and DD names - one for each of the two command log files. The sample started task EXBSTART delivered with the EXX109.JOBS data set uses DDCLOGR1 and DDCLOGR2 as default command log file names.

Linux

When starting a Broker with command log support, you must therefore specify two file names and paths - one for each of the two command log files. The sample startup script installed with the product uses file names `CMDLOG1` and `CMDLOG2` as the default command log file names.

Windows

When starting a Broker with command log support, you must therefore specify two file names and paths - one for each of the two command log files. The keys `ETB_CMDLOG1` and `ETB_CMDLOG2` are entered in the Registry with values `CMDLOGR1` and `CMDLOGR2` for the default command log file names.

At startup, Broker initializes both files and keeps one of them open. Command log statements are printed to the open file until the size of this file reaches the value specified in the Broker attribute `CMDLOG-FILE-SIZE`. This value must be specified in KB.

When the size of the open file exceeds the value specified in the Broker attribute `CMDLOG-FILE-SIZE`, Broker closes this file and opens the other, dormant file. Because the Broker closes a log file only when unable to print out a complete log line, the size of a *full* file may be smaller than `CMDLOG-FILE-SIZE`.

To switch log files on demand, using `etbcmd` | `ETBCMD`

- An open command log file can be forcibly closed even before the size limit is reached. Enter the following command.

  Under Linux:

  ```
  etbcmd -blocalhost:1970:TCP -cSWITCH-CMDLOG -dBROKER -xuser
  ```

  Under z/OS:

  ```
  //ETBCMD  EXEC PGM=ETBCMD,
  // PARM=('/-blocalhost:1970:TCP -cSWITCH-CMDLOG -xuser ',
  //       '-dBROKER')
  ```

  The command above will close the currently open file and open the one that has been dormant.

V Building an EntireX Broker Image

9 Building an EntireX Broker Image

- Prerequisites 132
- Building and Running the EntireX Broker Image 132
- Verifying the Build 136
- Healthcheck for EntireX Broker 137

Prerequisites

- Operating system Linux
- Docker installation 1.13.1 or compatible
- Software AG EntireX installation containing the packages
  - EntireX > Broker
  - EntireX > Adminstrating and Monitoring > Command Line Scripts

  See *EntireX Installation Packages* for full list.

Note: The current version of EntireX Broker Docker container supports only stateless scenarios (no use of persistent store).

Building and Running the EntireX Broker Image

The scripts provided with EntireX support the following three methods of building a Docker image and running the Docker container:

- Configuring with Modified Dockerfile
- Configuring during Image Start, using Default File Names
- Configuring during Image Start, using Custom File Names

Configuring with Modified Dockerfile

To copy the license and configuration files into the Docker image

1. Set your working directory to *<install_dir>/EntireX/docker/Broker*.
2. Create the TAR file containing all the necessary files with the following command:

```
./CreateEntireXBrokerTar.sh
```

3. Provide your configuration files into the current working directory, for example:

File	Req/ Opt
myLicense.xml	R
myEtbfile	O
myExxAppCert.pem	O
myExxAppKey.pem	O
myExxCACert.pem	O

4 Update the Dockerfile, for example:

```
# Possibility to add a valid license file already to the image instead of
# providing it during start up
# e.g.:
ADD myLicense.xml $EXXDIR/config/license.xml

# Possibility to add a different attribute file already to the image instead of
# providing it during start up
# e.g.:
ADD myEtbfile $EXXDIR/config/etb/$ETBIB/etbfile

# Possibility to add certificates for security broker
# e.g.:
ADD *.pem $EXXDIR/config/etb/$ETBID/
```

5 Build the EntireX Broker image, for example:

```
docker build -t exx_broker_image_1 .
```

With this method, the Docker build copies the configuration into the image. You will need to map your EntireX Broker ports during startup, for example:

```
docker run -d -p 2002:1971 -e ACCEPT_EULA=Y ↩
--name exx_broker_container_1 exx_broker_image_1
```

- **Advantages**
  Configuration changes can be persistent; you can reuse the configuration when a new version or fix is to be built. The complete configuration is in the image. For troubleshooting, Software AG Support will require only the image and the command you entered.
- **Disadvantage**
  If the configuration changes, you will need to build a new image and rerun the container.

Configuring during Image Start, using Default File Names

To copy the license and configuration files into container, using default file names

1 Set your working directory to *<install_dir>/EntireX/docker/Broker*.

2 Create the TAR file containing all the necessary files with the following command:

```
./CreateEntireXBrokerTar.sh
```

3 Build the EntireX Broker image, for example:

```
docker build -t exx_broker_image_2 .
```

4 Provide your configuration files with the default file names, for example:

File	Req/ Opt
license.xml	R
etbfile	O
exxAppCert.pem	O
exxAppKey.pem	O
exxCACert.pem	O

In this case the license and configuration files are mounted during startup. You will need to map your EntireX Broker ports during startup, for example:

```
docker run -d -p 2004:1971
                            -e ACCEPT_EULA=Y
                            -v <my-license-dir>:/licenses
                            -v <my-config-dir>:/configs
                            --name exx_broker_container_2 exx_broker_image_2
```

- **Advantages**
  Configuration changes can be persistent; if the configuration changes, you only need to rerun the container.
- **Disadvantage**
  The configuration is in the image and in the configuration files mounted to the container. For troubleshooting, Software AG Support will require an image, configuration files and the command you entered.

Configuring during Image Start, using Custom File Names

To copy the license and configuration files into container, using custom file names

1 Set your working directory to *<install_dir>/EntireX/docker/Broker*.

2 Create the TAR file containing all the necessary files with the following command:

```
./CreateEntireXBrokerTar.sh
```

3 Build the EntireX Broker image, for example:

```
docker build -t exx_broker_image_3 .
```

4 Provide your configuration files with the custom file names, for example:

File	Req/ Opt
<my-license-dir>/myLicense.xml	R
<my-config-dir>/myEtbfile	O
<my-config-dir>/myExxAppCert.pem	O
<my-config-dir>/myExxAppKey.pem	O
<my-config-dir>/myExxCACert.pem	O

In this case the license and configuration files are mounted during startup. License file and etbfile will be renamed to match EntireX Broker naming conventions. You will need to map your EntireX Broker ports during startup, for example:

```
docker run -d -p 2004:1971
              -e ACCEPT_EULA=Y
              -e "EXX_ATTRIBUTE=myEtbfile"
              -e "EXX_LICENSE_KEY=myLicense.xml"
              -e "EXX_KEY_FILE=myExxAppKey.pem"
              -e "EXX_KEY_STORE=myExxAppCert.pem"
              -e "EXX_TRUST_STORE=myExxCACert.pem"
              -v <my-license-dir>:/licenses
              -v <my-config-dir>:/configs
              --name exx_broker_container_3 exx_broker_image_3
```

- **Advantages**
  Configuration changes can be persistent; you are free to choose your own file names. If the configuration changes, you only need to to rerun the container.

- **Disadvantage**
  The configuration is in the image and in the configuration files mounted to the container. For troubleshooting, Software AG Support will require an image, configuration files and the command you entered.

Verifying the Build

To verify the build

1 Show the image with command

```
docker images
```

2 Start the docker image to be verified as described above, for example:

```
docker run -d -p 2001:1971 -e ACCEPT_EULA=Y ↩
--name exx_broker_container_1 exx_broker_image_1
```

3 Show the log:

```
docker logs -f exx_broker_container_1
```

4 Show the containers:

```
docker ps
```

5 Stop the container:

```
docker stop exx_broker_container_1
```

6 Delete the container:

```
docker rm exx_broker_container_1
```

7 Remove the image:

```
docker rmi exx_broker_image_1
```

Healthcheck for EntireX Broker

The *docker* directory for EntireX Broker contains a script `healthcheck.sh`. Execution of this script pings the broker and returns the result of the ping command:

0	success
all other values	ping failure

In the Docker context, this `healthcheck.sh` is put into the Docker container and enabled by setting the `HEALTHCHECK` instruction in the Dockerfile.

You can also use the `healthcheck.sh` script in the context of an orchestration tool (e.g. Kubernetes) to enable healthcheck functionality.