

webMethods EntireX

Broker

Version 10.1

October 2017

This document applies to webMethods EntireX Version 10.1 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 © 1997-2017 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-101-20191129

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	10
3 General Architecture of EntireX Broker	13
Introduction to EntireX Broker Architecture	14
Client Server Communication Model	15
Architecture of Broker Stub	18
Architecture of Broker Kernel	20
4 Functionality of EntireX Broker	23
Application Bindings (Stubs)	24
Attach Services	25
Character Conversion	25
Command and Information Services	26
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	66
Codepage-specific Attributes	79
Adabas SVC/Entire Net-Work-specific Attributes	82
Security-specific Attributes	85
TCP/IP-specific Attributes	92
c-tree-specific Attributes	95
SSL/TLS-specific Attributes	97
DIV-specific Attributes	102
Adabas-specific Attributes	103
Application Monitoring-specific Attributes	105

Authorization Rule-specific Attributes	106
Variable Definition File	107
III Broker Command and Information Services	109
7 Broker Command and Information Services	111
CIS Overview Table	112
Modes of Requesting the Services	113
ETBCMD: Executable Command Requests	115
ETBINFO: Returnable Information Requests	120
IV Using Sample Security Exits for Broker Security	121
8 Using Sample Security Exits for Broker Security	123
Overview of Security Data Flow	124
Prerequisites for Running EntireX Broker in a Secure Environment	125
General Security Recommendations	125
Writing Security Exits	127
Security-Related Parameters	129
Programming Broker Stub Exits	131
Required/Recommended Actions in the Exit (depending on Security Type)	135
V	137
9 EntireX Broker Reporting	139
Configuration Report	140
Load Module Report	141
Storage Report	142
Persistent Store Report	145
License Report	147
10 Command Logging in EntireX	149
Introduction to Command Logging	150
Command Log Filtering using Command-line Interface ETBCMD	152
ACI-driven Command Logging	154
Dual Command Log Files	154

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 <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	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

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <http://documentation.softwareag.com>. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>.

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.asp and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

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.

<i>Concept of Interoperability</i>	Introduces the basic concept of EntireX Broker: achieving highly flexible interoperability of distributed application components.
<i>General Architecture of EntireX Broker</i>	Describes the components and transport mechanisms of EntireX Broker within the context of EntireX.
<i>Functionality of EntireX Broker</i>	Provides a brief overview of the functionality provided by EntireX Broker.
<i>Broker Quick Reference</i>	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	10

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

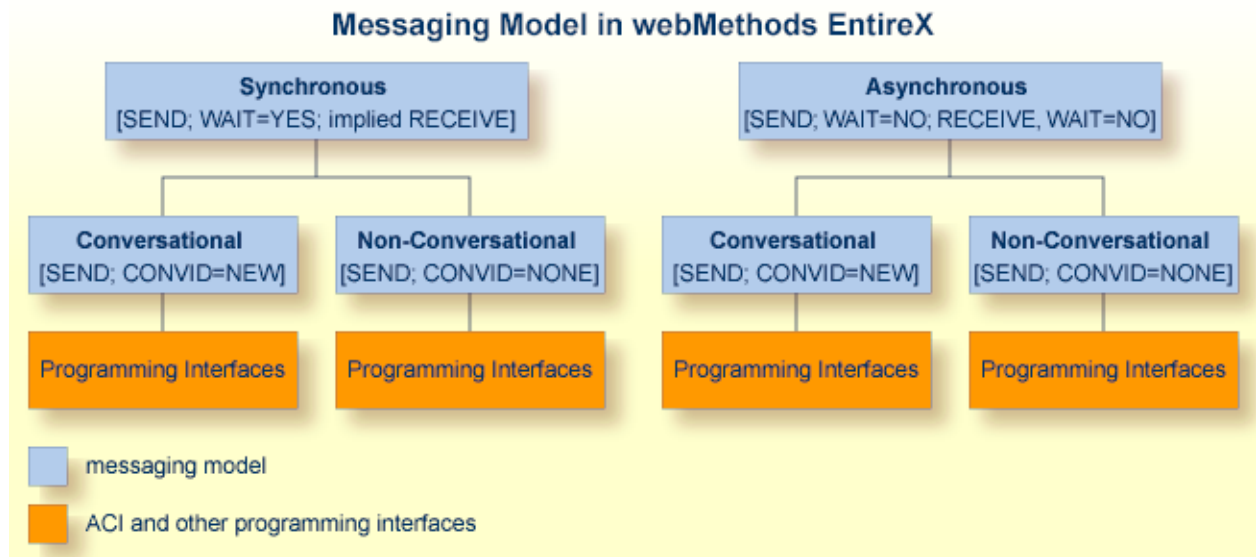
In a distributed processing environment that uses EntireX Broker, communication occurs through application components exchanging messages. An application component offering a service registers it with EntireX Broker (see REGISTER); this makes the service available to other application components able 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

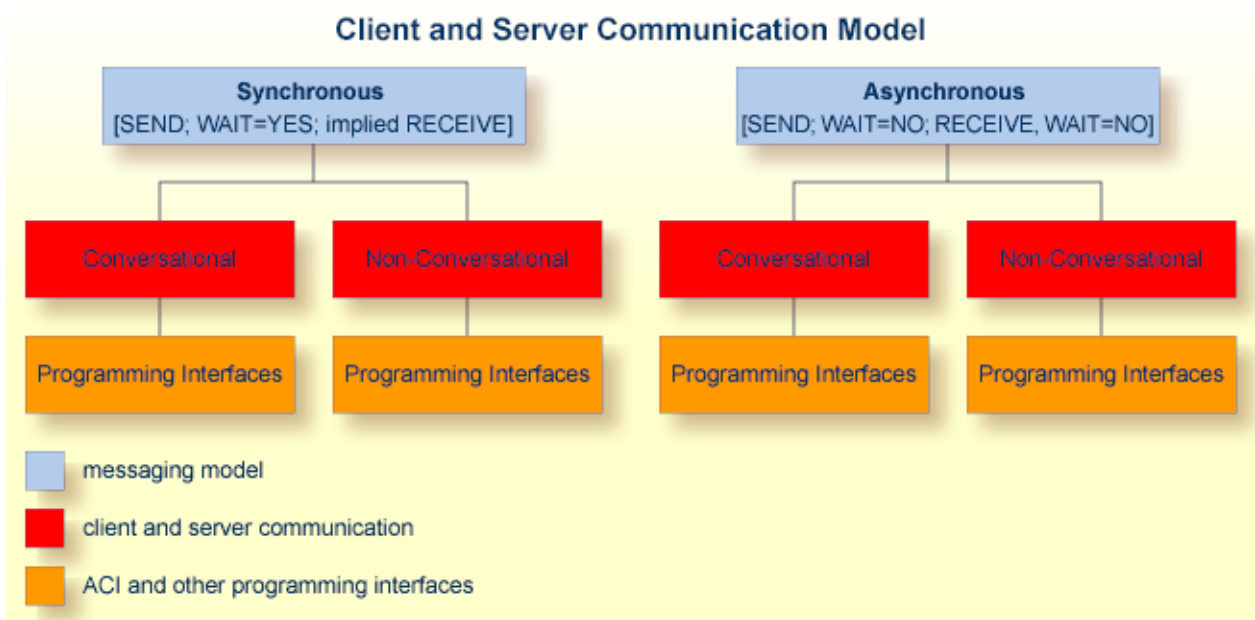


Notes:

1. The synchronous SEND, WAIT=YES command contains an implied RECEIVE command.
2. Persistence available. See *Concepts of Persistent Messaging*.

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	15
■ Architecture of Broker Stub	18
■ Architecture of Broker Kernel	20

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.

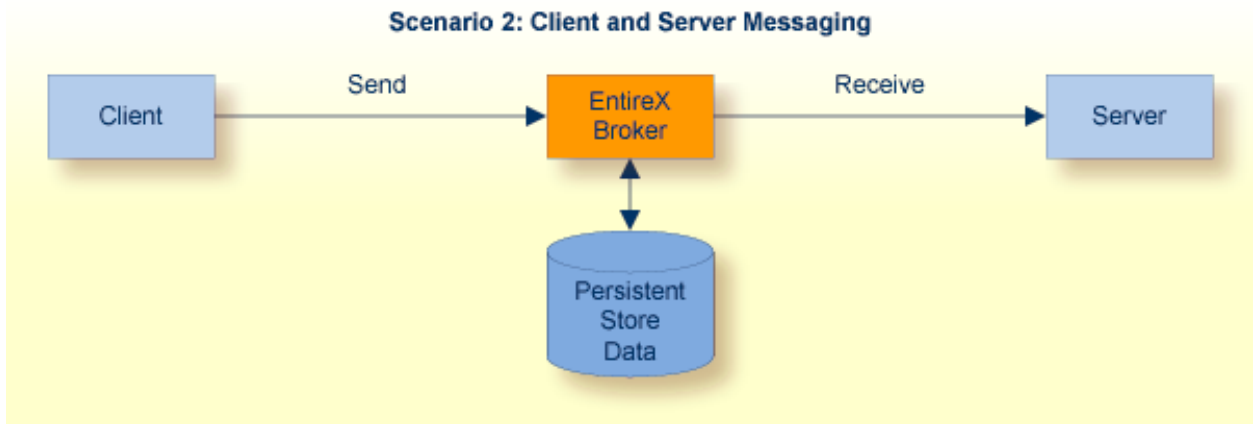
Client Server Communication Model

The EntireX Broker uses the communication model client and server. See *Writing Client and Server Applications* 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: Client and server have specific meanings within the context of EntireX.

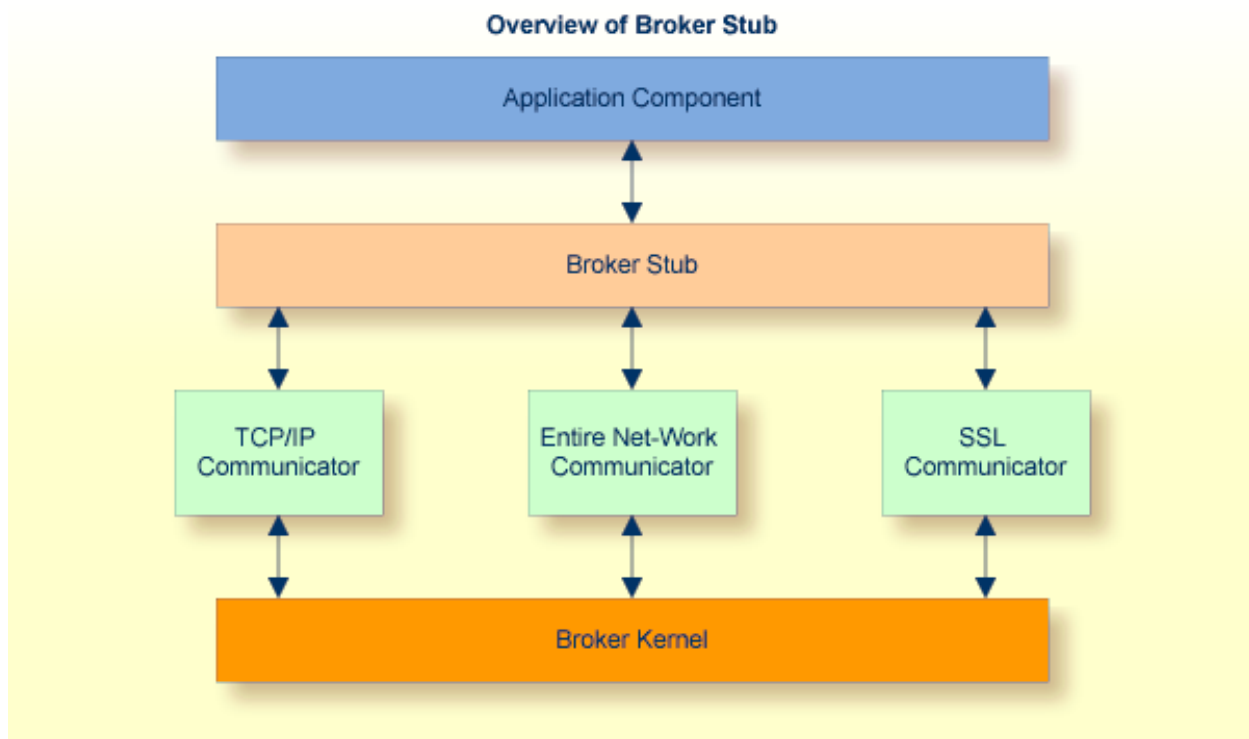
Term	Description
Client	<p>An application component intending to access a service makes its request via EntireX Broker which routes the request to the specific application component offering this service.</p> <p>The request can be a single pair of messages comprising request/reply; or it can be a sequence 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.</p>
Server	<p>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.</p>

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

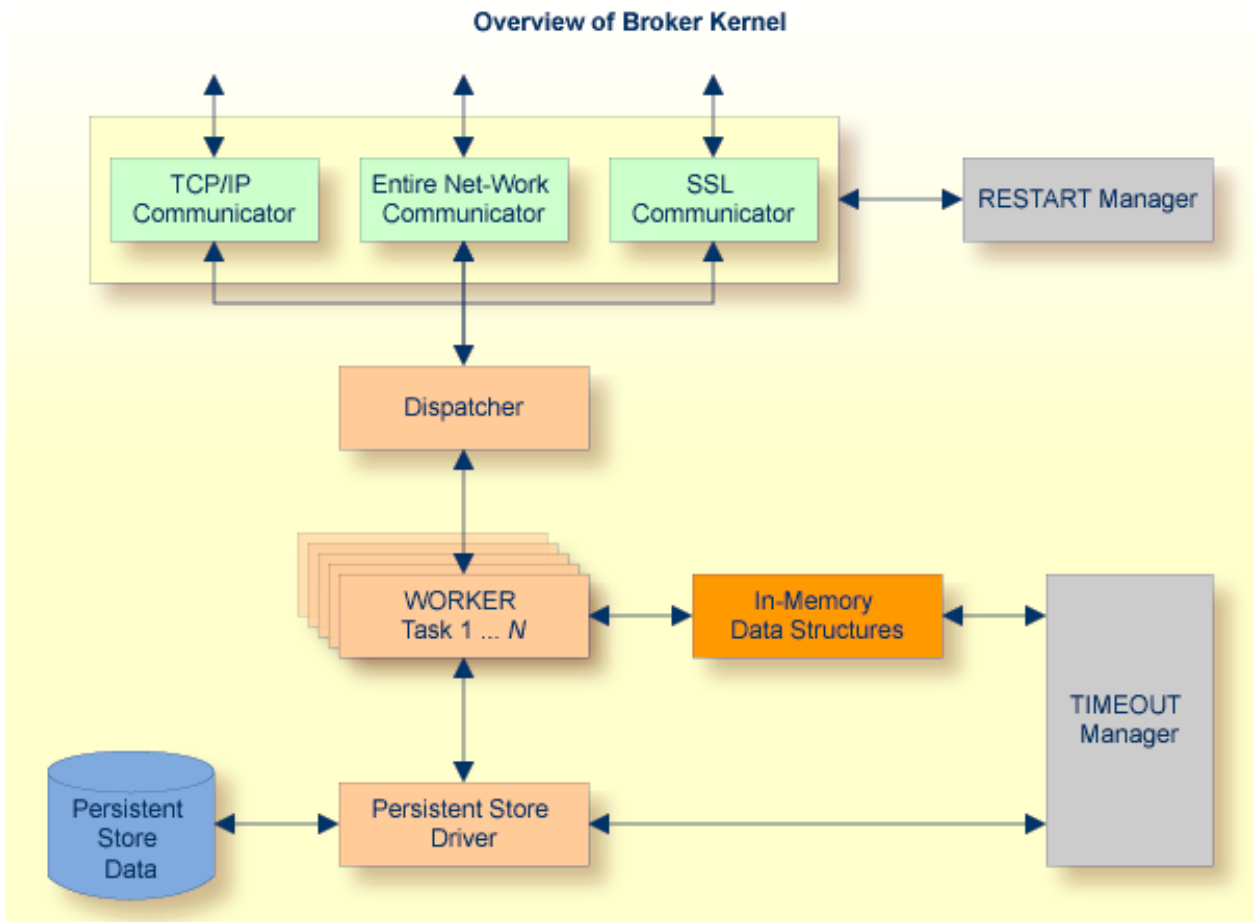
The following table gives a step-by-step description of a typical command process flow from and to a Broker stub. This example describes a `SEND/RECEIVE` command pair.

Step	Description
1	The originating application program calls the stub with a <code>SEND/WAIT=YES</code> 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 <code>SEND, WAIT=YES</code> 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 <code>SEND</code> 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 <code>RECEIVE</code> 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
■ Attach Services	25
■ Character Conversion	25
■ Command and Information Services	26
■ 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)

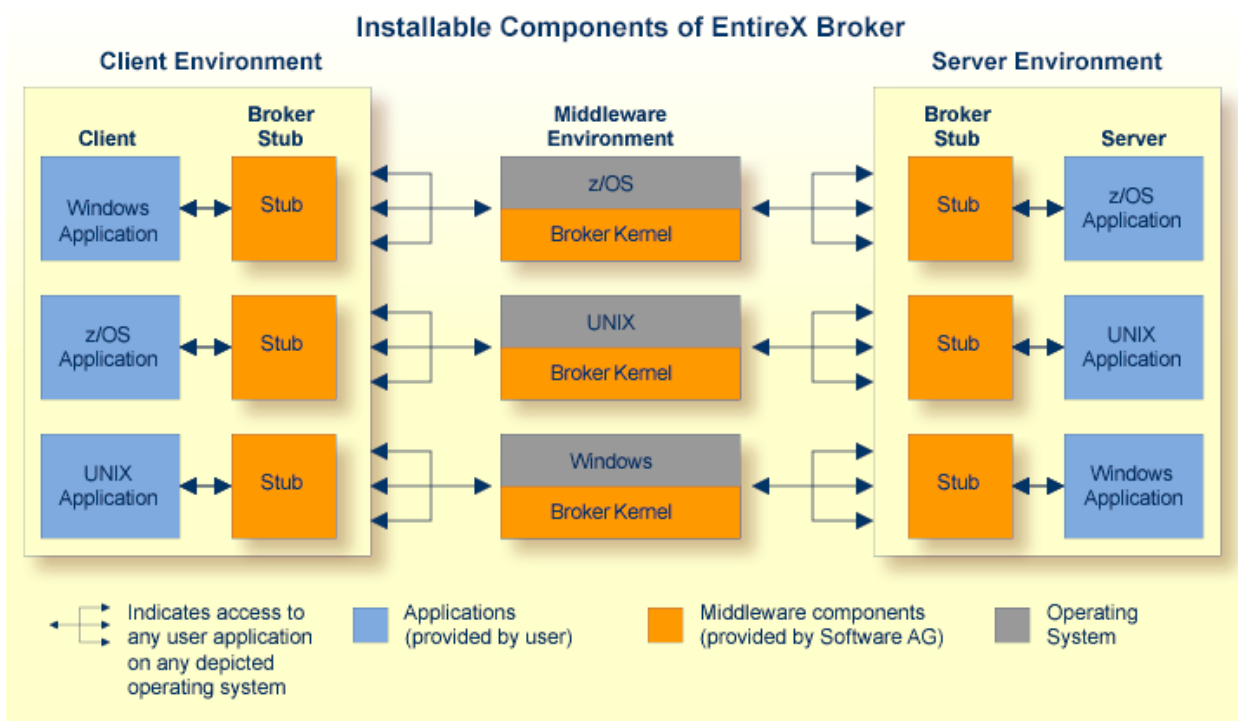
Application bindings allow applications developed in different programming languages and executing on various different platforms to be enabled by using EntireX Broker, see [Architecture of Broker Stub](#). Specifically, Java, Natural and other programs are easily enabled using EntireX Broker. These bindings are available on all major mainframe, UNIX 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, UNIX, Windows etc. communicating with each other using stubs:



Attach Services

EntireX Broker provides a choice of mechanisms which enable application components to be started automatically when required.

Example: A client application requires some processing from a server application component. The range of attach services includes starting IMS TM and CICS transactions on the mainframe, and batch programs/processes on mainframe, UNIX and Windows.

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

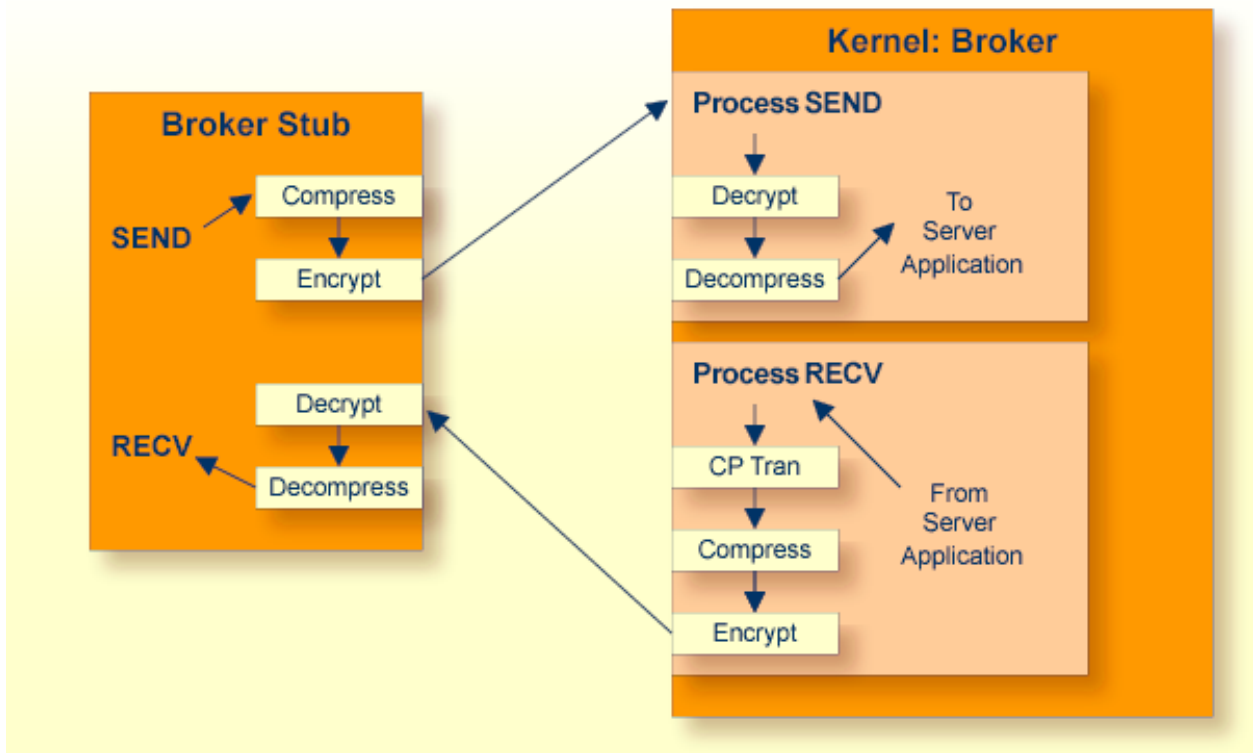
EntireX Broker provides accounting information based upon the flow of message sequences (or conversations). On z/OS, this information is written to standard accounting (SMF) records; on other platforms it is written to a file. The information can be used for:

- 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 [EntireX 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 [EntireX 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.	UNIX, Windows, z/OS, z/VSE	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.	UNIX 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*.

Security

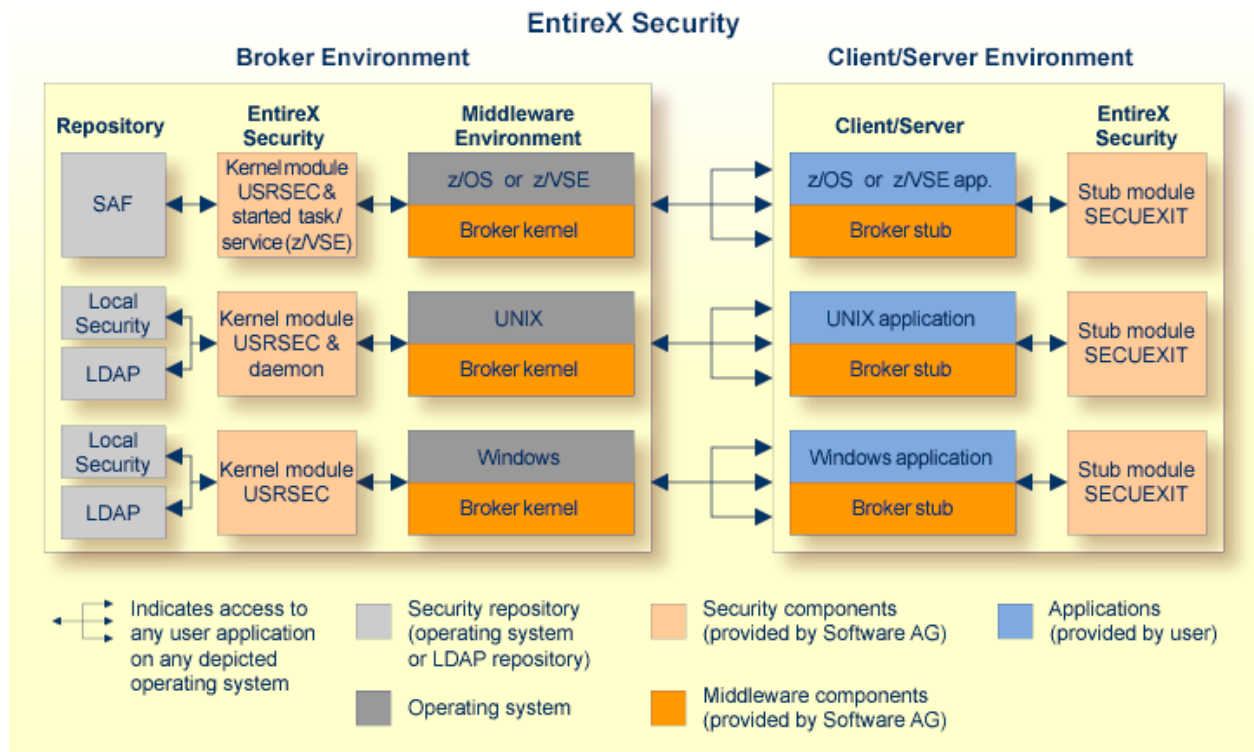
EntireX Security enables distributed application components running with Broker to be executed securely. EntireX Security is located centrally in the kernel of EntireX Broker giving it an overview of all messages sent between application components and therefore providing complete control over the authentication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

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

- RACF
- CA ACF2
- CA Top Secret
- UNIX 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 *Security Solutions in EntireX*.

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

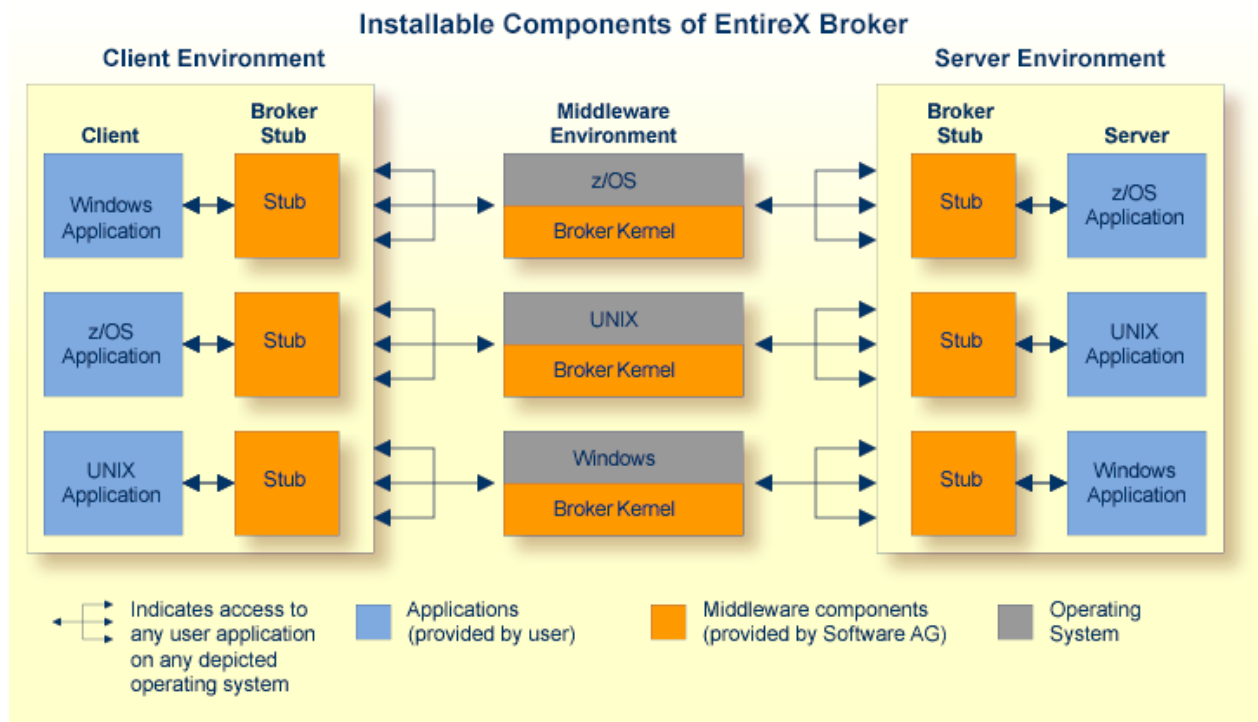


Notes:

1. The synchronous SEND, WAIT=YES command contains an implied RECEIVE command.
2. Persistence available. See *Concepts of Persistent Messaging*.

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	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	
	UNIX System Services	<i>EntireX Java ACI</i>	x	x		x
UNIX		broker.so	x	x		
		<i>EntireX Java ACI</i>	x	x		x
Windows		broker.dll ⁽⁴⁾	x	x		
		<i>EntireX Java ACI</i>	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 UNIX 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 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 *Setting up and Administering the EntireX Broker HTTP(S) Agent* in the UNIX and Windows 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 and Certificates with EntireX*.

See also:

- *Transport Methods for Broker Stubs* in the platform-specific broker stub 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	66
▪ Codepage-specific Attributes	79
▪ Adabas SVC/Entire Net-Work-specific Attributes	82
▪ Security-specific Attributes	85
▪ TCP/IP-specific Attributes	92
▪ c-tree-specific Attributes	95
▪ SSL/TLS-specific Attributes	97
▪ DIV-specific Attributes	102
▪ Adabas-specific Attributes	103
▪ Application Monitoring-specific Attributes	105
▪ Authorization Rule-specific Attributes	106
▪ Variable Definition File	107



Note: This section lists all EntireX Broker parameters. Not all parameters are applicable to all supported operating systems.

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.

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 <i>EXBATTR</i> in the EntireX Broker source library.
UNIX	File <i>etbfile</i> in directory <i><InstDir>/EntireX/config/etb/<BrokerName></i> (default) *
Windows	File <i><BrokerName>.atr</i> in directory <i><InstDir>\EntireX\config\etb\<BrokerName></i> (default) *
BS2000	File <i>ETB-ATTR</i> in library <i>EXX101.JOBS</i> .
z/VSE	Library member <i>ETBnnn.ATR</i> , where <i>nnn</i> is a placeholder specifying the broker instance (e.g. <i>nnn</i> =the assigned broker ID).

* 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 (UNIX 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 UNIX and Windows, the string `${variable name}` is replaced with the value of the corresponding environment variable.
 - On z/OS, variable values are read from a file defined by the DD name `ETBVAR`. The syntax of this file is the same as the attribute file.
 - If a variable has no value: if the variable name is enclosed in braces, error 00210594 is given, otherwise `$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	UNIX	Windows	z/VSE	BS2000
ABEND-LOOP-DETECTION	<u>YES</u> NO	O	z	u	w	v	b
	<p>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.</p> <p>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.</p>						
ABEND-MEMORY-DUMP	<u>YES</u> NO	O	z	u	w	v	b
	<p>YES Print all data pools of the broker if a task terminates abnormally. This dump is needed to analyze the abend.</p> <p>NO If the dump has already been sent to Software AG, you can set to "NO" to avoid the extra overhead.</p>						
ACCOUNTING	<u>NO</u> 128-255	O	z				
	<u>NO</u> YES [SEPARATOR= <i>char</i>]	O		u	w	v	b
	<p>Determines whether accounting records are created.</p> <p>NO Do not create accounting records.</p> <p><i>nnn</i> The SMF record number to use when writing the accounting records.</p> <p>YES Create accounting data.</p> <p><i>char</i>= separator character(s). Up to seven separator characters can be specified using the SEPARATOR suboption, for example</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	ACCOUNTING = (YES, SEPARATOR=;). If no separator character is specified, the comma character will be used. See also <i>Accounting in EntireX Broker</i> in the platform-specific Administration documentation.						
ACCOUNTING-VERSION	<u>1</u> 2 3 4 5	O	z	u	w	v	b
	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.						
APPLICATION-MONITORING or APPMON	YES <u>NO</u>	O	z	u	w	v	
	Enable application monitoring in EntireX Broker. YES Enable application monitoring. NO Disable application monitoring. See <i>Application Monitoring</i> .						
AUTOLOGON	<u>YES</u> NO	O	z	u	w	v	b
	YES LOGON occurs automatically during the first SEND or REGISTER. NO The application has to issue a LOGON call.						
BLACKLIST-PENALTY-TIME	<u>5m</u> n nS nM nH	R	z	u	w	v	b
	Define the length of time a participant is placed on the PARTICIPANT-BLACKLIST to prevent a denial-of-service attack. n Same as nS. nS Non-activity time in seconds (max. 2147483647). nM Non-activity time in minutes (max. 35791394).						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<i>n</i> H Non-activity time in hours (max. 596523). See <i>Protecting a Broker against Denial-of-Service Attacks</i> in the platform-specific broker Administration documentation.						
BROKER-ID	A32	R	z	u	w	v	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	<u>15M</u> <i>n</i> <i>nS</i> <i>nM</i> <i>nH</i>	R	z	u	w	v	b
	Define the non-activity time for clients. <i>n</i> Same as <i>nS</i> . <i>nS</i> Non-activity time in seconds (max. 2147483647). <i>nM</i> Non-activity time in minutes (max. 35791394). <i>nH</i> 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	<u>NO</u> YES	O	z	u	w	v	b
	NO Command logging will not be available in the broker. YES Command logging features will be available in the broker.						
CMDLOG-FILE-SIZE	<u>1024</u> <i>n</i>	O	z	u	w	v	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	<u>60s</u> <i>n</i> <i>nS</i> <i>nM</i> <i>nH</i>	O	z	u	w	v	b
	Defines the time interval of time-driven broker-to-broker calls. 1. It controls the time between handshake attempts.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	2. The standby broker will check the status of the standard broker after the elapsed CONTROL - INTERVAL time. <i>n</i> Same as <i>nS</i> . <i>nS</i> Interval in seconds (max. 2147483647). <i>nM</i> Interval in minutes (max. 35791394). <i>nH</i> 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	<u>UNLIM</u> <i>n</i>	O	z	u	w	v	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. <i>n</i> Number of conversations. This value can be overridden by specifying a CONV - LIMIT for the service. A value of 0 (zero) is invalid.						
DEFERRED	<u>NO</u> YES	O	z	u	w	v	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	<u>YES</u> NO	O	z	u	w	v	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 <i>Dynamic Memory Management</i> under <i>Broker Resource Allocation</i> . 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	UNIX	Windows	z/VSE	BS2000
	<p>If you run your broker with attribute <code>DYNAMIC-MEMORY-MANAGEMENT=YES</code>, the following attributes are not needed:</p> <div><div>■ <code>CONV-DEFAULT</code></div><div>■ <code>NUM-SERVER</code></div><div>■ <code>HEAP-SIZE</code></div><div>■ <code>NUM-SERVICE-EXTENSION</code></div><div>■ <code>LONG-BUFFER-DEFAULT</code></div><div>■ <code>NUM-SERVICE</code></div><div>■ <code>SERVER-DEFAULT</code></div><div>■ <code>NUM-SHORT[-BUFFER]</code></div><div>■ <code>SHORT-BUFFER-DEFAULT</code></div><div>■ <code>NUM-UOW MAX-UOWS MUOW</code></div><div>■ <code>NUM-CLIENT</code></div><div>■ <code>NUM-WQE</code></div><div>■ <code>NUM-CMDLOG-FILTER</code></div><div>■ <code>NUM-COMBUF</code></div><div>■ <code>NUM-CONV[ERSATION]</code></div><div>■ <code>NUM-LONG[-BUFFER]</code></div></div> <p>Caution: However, if one of these attributes is defined, it determines the allocation size of that particular broker resource.</p>						
DYNAMIC-WORKER-MANAGEMENT	<u>NO</u> YES	O	z	u	w		b
	<p>NO All worker tasks are started at broker startup. The number of worker tasks is defined by <code>NUM-WORKER</code>. 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.</p> <p>YES As above, the initial portion of worker tasks started at broker startup is determined by <code>NUM-WORKER</code>. 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 <code>WORKER-MIN</code> and <code>WORKER-MAX</code>.</p> <p>If you run broker with <code>DYNAMIC-WORKER-MANAGEMENT=YES</code>, the following attributes are useful to optimize the overall processing:</p> <div>■ <code>WORKER-MAX</code></div> <div>■ <code>WORKER-MIN</code></div> <div>■ <code>WORKER-NONACT</code></div> <div>■ <code>WORKER-QUEUE-DEPTH</code></div>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<div>■ WORKER-START-DELAY</div> <p>The attribute NUM-WORKER defines the initial number of worker tasks started during initialization. See <i>Dynamic Worker Management</i> under <i>Broker Resource Allocation</i>.</p>						
FORCE	<u>NO</u> YES	O		u			
	<p>NO Go down with error if IPC resources still exist.</p> <p>YES Clean up the left-over IPC resources of a previous run.</p> <p>Note:</p> <ol style="list-style-type: none">1. If broker is started twice, the second instance will kill the first by removing the IPC resources.2. For BS2000, z/OS and z/VSE, see separate attribute FORCE under DEFAULTS=NET.						
HEAP-SIZE	<u>1024</u> <i>n</i>	O	z	u	w	v	b
	Defines the size of the internal heap in KB. Not required if you are using DYNAMIC-MEMORY-MANAGEMENT. If you are <i>not</i> using dynamic memory management, we strongly recommend specifying - as a minimum - the default value of 1024 KB.						
ICU-CONVERSION	<u>YES</u> NO	O	z	u	w	v	b
	<p>Disable or enable ICU conversion. Default for z/VSE is NO; other platforms YES.</p> <p>YES ICU is loaded and available for conversion. It is a prerequisite for CONVERSION=SAGTCHA and CONVERSION=SAGTRPC.</p> <p>NO ICU is not loaded and not available for conversion. CONVERSION=SAGTCHA and CONVERSION=SAGTRPC cannot be used.</p> <p>If any of the broker service definitions uses the character conversion approach <i>ICU Conversion</i>, that is, CONVERSION=SAGTCHA or CONVERSION=SAGTRPC, ICU-CONVERSION must be set to YES. If you are using only a user exit (see <i>User Exits</i>) or CONVERSION=NO as character conversion approach for all your broker service definitions, ICU-CONVERSION can be set to "NO".</p> <p>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.</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
ICU-DATA-DIRECTORY	Folder or directory name in quotes.	O	z	u	w		
	The location where the broker searches for ICU custom converters. See <i>Building and Installing ICU Custom Converters</i> in the platform-specific Administration documentation.						
ICU-SET-DATA-DIRECTORY	<u>YES</u> 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 <i>Building and Installing ICU Custom Converters</i> in the platform-specific Administration documentation. NO Use of ICU custom converters is not possible.						
IPV6	YES <u>NO</u>	O	z	u	w		b
	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	<u>UNLIM</u> <i>n</i>	O	z	u	w	v	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. <i>n</i> 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	<u>0</u> <i>n</i> <i>nK</i> <i>nM</i> <i>nG</i> UNLIM	O	z	u	w	v	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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
MAX-MESSAGE-LENGTH	<u>2147483647</u> <i>n</i>	O	z	u	w	v	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	<u>16</u> <i>n</i>	O	z	u	w	v	b
	Maximum number of messages in a UOW.						
MAX-MSG	See MAX-MESSAGE-LENGTH.						
MAX-TRACE-FILES	<u>4</u> <i>n</i>	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 <i>Trace File Handling</i> in the UNIX and Windows Administration documentation.						
MAX-UOW-MESSAGE-LENGTH	See MAX-MESSAGE-LENGTH.						
MAX-UOWS	<u>0</u> <i>n</i>	O	z	u	w	v	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. 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	<u>NONE</u> UPPER LOWER	O	z	u	w	v	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	<u>YES</u> NO	O	z	u	w	v	b
	YES New UOW messages are allowed. NO New UOW messages are not allowed.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>This applies to UOW when using Persistence and should not be used for non-persistent UOWs. A usage example could be the following:</p> <p>The broker persistent store reaches capacity and the broker shuts down. You can set <code>NEW-UOW-MESSAGES</code> 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 <code>ALLOW-NEWUOWMSGs</code>. This action allows new UOW messages to be sent to the broker. Reset attribute <code>NEW-UOW-MESSAGES</code> to "YES", which permits new UOW messages to be produced in subsequent broker sessions.</p>						
NUM-BLACKLIST-ENTRIES	<u>256</u> <i>n</i>	O	z	u	w	v	b
	Number of entries in the participant blacklist. Default value is 256 entries. Together with <code>BLACKLIST-PENALTY-TIME</code> and <code>PARTICIPANT-BLACKLIST</code> , this attribute is used to protect a broker running with <code>SECURITY=YES</code> against denial-of-service attacks. See <i>Protecting a Broker against Denial-of-Service Attacks</i> in the platform-specific broker Administration documentation.						
NUM-CLIENT	<i>n</i>	R	z	u	w	v	b
	Number of clients that can access the broker concurrently. A value of 0 (zero) is invalid.						
NUM-CMDLOG-FILTER	<u>1</u> <i>n</i>	O	z	u	w	v	b
	<p>Maximum number of filters that can be specified simultaneously.</p> <p>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 <code>CMDLOG</code> is set to "YES". See Command Logging in EntireX for more information.</p>						
NUM-COMBUF	<u>1024</u> 1 - 999999	R	z	u	w	v	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	<i>n</i> AUTO	R	z	u	w	v	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.)						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p><i>n</i> Number of conversations.</p> <p>AUTO Uses the CONV-DEFAULT and the service-specific CONV-LIMIT values to calculate the number of conversations. The values used in the calculation must not be set to "UNLIM".</p> <p>Note:</p> <p>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.</p> <p>2. See <i>Wildcard Service Definitions</i> under <i>Broker Attributes</i>.</p>						
NUM-LONG-BUFFER or NUM-LONG	<u>4096</u> <i>n</i> AUTO	R	z	u	w	v	b
	<p>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.</p> <p><i>n</i> Number of buffers.</p> <p>AUTO Uses the LONG-BUFFER-DEFAULT and the service-specific LONG-BUFFER-LIMIT values to calculate the number of long message buffers. The values used in the calculation must not be set to "UNLIM".</p> <p>A value of 0 (zero) is invalid.</p> <p>In <i>non-conversational</i> 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.</p> <p>In <i>conversational</i> mode, the last message received is always kept until a new one is received.</p> <p>Note:</p> <p>1. If a catch-all service is defined in the service-specific section of the attribute file, the value of AUTO is invalid.</p> <p>2. See <i>Wildcard Service Definitions</i> under <i>Broker Attributes</i>.</p>						
NUM-PARTICIPANT-EXTENSION	<i>n</i>	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
	Defines the number of participant extensions to link participants as clients and servers. <i>n</i> Number of participant extensions <i>not specified</i> 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	<i>n</i> AUTO	R	z	u	w	v	b
	Defines the number of servers that can offer services concurrently using the broker. This is <i>not</i> the number of services that can be registered to the broker (see NUM-SERVICE). <i>n</i> Number of servers. AUTO Uses the SERVER-DEFAULT and the service-specific SERVER-LIMIT values to calculate the number of servers. The values used in the calculation must not be set 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 <i>Wildcard Service Definitions</i> under <i>Broker Attributes</i> .						
NUM-SERVICE	<i>n</i>	R	z	u	w	v	b
	Defines the number of services that can be registered to the broker. This is <i>not</i> the number of servers that can offer the services (see NUM-SERVER). A value of 0 (zero) is invalid.						
NUM-SERVICE-EXTENSION	<i>n</i> AUTO	O	z	u	w	v	b
	Defines the number of service extensions to link servers to services. <i>n</i> Number of service extensions. AUTO Uses the value specified or calculated for NUM-SERVER + NUM-CLIENT, plus an extra cushion. <i>not specified</i> If this attribute is not set, the default value is NUM-SERVER multiplied by NUM-SERVICE.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>The minimum value is NUM-SERVER. The maximum value is NUM-SERVER multiplied by NUM-SERVICE.</p> <p>Caution is recommended with this attribute:</p> <ul style="list-style-type: none">■ Set this attribute only if the storage resources allocated for service extensions need to be restricted.■ Note that the value $\langle n \rangle$ allows only the specified number of server instances of $\langle n \rangle$ 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 (see note below).						
NUM-SHORT-BUFFER or NUM-SHORT	n AUTO	R	z	u	w	v	b
	<p>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.</p> <p>n Number of buffers.</p> <p>AUTO Uses the SHORT-BUFFER-DEFAULT and the service-specific SHORT-BUFFER-LIMIT values to calculate the number of short message buffers. The values used in the calculation must not be set to "UNLIM".</p> <p>Note:</p> <ol style="list-style-type: none">1. In <i>non-conversational</i> 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 <i>conversational</i> 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 <i>Wildcard Service Definitions</i> under <i>Broker Attributes</i>.						
NUM-UOW	Q n	O	z	u	w	v	b
	<p>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</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
	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	<u>1</u> <i>n</i> (max. 10)	R	z	u	w	v	b
	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	v	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	<u>YES</u> NO	R	z	u	w	v	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 <i>Protecting a Broker against Denial-of-Service Attacks</i> in the platform-specific broker Administration documentation.						
PARTNER-CLUSTER-ADDRESS	A32	R	z	u	w	v	b
	This is the address of the load/unload broker in transport-method-style. Transport methods TCP and SSL are supported. See <i>Transport-method-style Broker ID</i> for more details. This attribute is required if the attribute RUN-MODE is specified.						
PERCENTAGE-FOR-CONNECTION-SHORTAGE-MESSAGE	<u>90</u> 1-100	O	z	u	w	v	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 <u>NO</u>	O	z	u		v	
	In earlier EntireX versions, the maximum number of TCP/IP connections per communicator was limited; see <i>Maximum TCP/IP Connections per</i>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p><i>Communicator</i> under <i>Broker Resource Allocation</i> for platform-specific list. With attribute <code>POLL</code> introduced in EntireX version 9.0, this restriction can be lifted under z/OS, UNIX and z/VSE.</p> <p>NO This setting is used to run the compatibility mode in Broker. The <code>poll()</code> system call is not used. The limitations described under <i>Maximum TCP/IP Connections per Communicator</i> under <i>Broker Resource Allocation</i> apply.</p> <p>YES The <code>poll()</code> system call is used to lift the resource restrictions with <code>select()</code> in multiplexing file descriptor sets.</p> <p>Note: Setting this attribute to YES increases CPU consumption. <code>POLL=YES</code> is only useful if you need more than the maximum number of TCP/IP connections per communicator; we recommend <code>POLL=NO</code> to reduce CPU consumption.</p>						
PSTORE	<u>NO</u> HOT COLD	O	z	u	w	v	b
	<p>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", <code>PSTORE - TYPE</code> must be set.</p> <p>NO No persistent store.</p> <p>HOT Persistent UOWs are restored to their prior state during initialization.</p> <p>COLD Persistent UOWs are not restored during initialization, and the persistent store is considered empty.</p> <p>Note: For a hot or cold start, the persistent store must be available when your broker is restarted.</p>						
PSTORE - REPORT	<u>NO</u> YES	O	z	u	w	v	b
	<p>Determines whether PSTORE report is created.</p> <p>NO Do not create the PSTORE report file.</p> <p>YES Create the PSTORE report file.</p> <p>See also <i>Persistent Store Report</i>.</p>						
PSTORE - TYPE	DIV (z/OS) CTREE (UNIX, Windows) Adabas (all platforms) FILE (UNIX, Windows)	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
	Describes the type of persistent store driver required.						
	DIV Data in Virtual. z/OS only, and default on this platform. See <i>DIV-specific Attributes</i> below and <i>Implementing a DIV Persistent Store</i> under <i>Managing the Broker Persistent Store</i> .						
	CTREE c-tree database. UNIX and Windows only. See <i>c-tree-specific Attributes</i> and <i>c-tree Database as Persistent Store</i> in the UNIX and Windows Administration documentation.						
	ADABAS Adabas. All platforms. See also <i>Adabas-specific Attributes</i> (below) and <i>Managing the Broker Persistent Store</i> in the platform-specific Administration documentation.						
	FILE B-Tree database. UNIX and Windows only. No longer supported.						
PSTORE-VERSION	2 3 4 5	O	z	u	w	v	b
	Determines the version of the persistent store. PSTORE=COLD is not needed to upgrade the PSTORE to version 3. Any broker restart with PSTORE-VERSION=3 will upgrade the PSTORE version.						
	PSTORE-VERSION=3 is needed for ICU support.						
	The DIV PSTORE requires PSTORE-VERSION=4.						
	PSTORE-VERSION=5 was added in EntireX version 10.1 to support 64-bit time values on z/OS, and unique message IDs on all platforms. See <i>Unique Message ID</i> . PSTORE-VERSION=5 significantly improvement Adabas PSTORE performance on all platforms. We strongly recommend you use this version.						
	Caution:						
	■ If you go back to PSTORE-VERSION=2 after upgrading to PSTORE-VERSION=3, the broker will only process data previously created with version 2. No version 3 data will be accessible.						
	■ If you change the DIV PSTORE from version 3 to 4, perform a COLD restart for the change to take effect, or run PSTORE UNLOAD/LOAD first.						
	■ If you change to PSTORE-VERSION=5, perform a COLD restart for the change to take effect.						
RUN-MODE	STANDARD STANDBY PSTORE-LOAD PSTORE-UNLOAD	O	z	u	w	v	b
	Determines the initial run mode of the broker.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<div>STANDARD Default value. Normal mode.</div> <div>STANDBY Deprecated. Supported for compatibility reasons.</div> <div>PSTORE-LOAD Broker will run as load broker to write Persistent Store data to a new persistent store. See also <i>Migrating the Persistent Store</i>.</div> <div>PSTORE-UNLOAD Broker will run as unload broker to read an existing persistent store and pass the data to a broker running in PSTORE-LOAD mode. See also <i>Migrating the Persistent Store</i>.</div>						
SECURITY	<div>NO YES</div> <div>Determines whether the EntireX Broker security exits are activated.</div> <div>NO The security exits are not activated.</div> <div>YES The security exits are activated. If the security routines cannot be activated, the broker will not start.</div> <div>Broker trace reports the type of security which is active and from where the security module USRSEC is loaded:</div> <div><div>■ EntireX Security</div><div>■ User-written USRSEC.</div></div>	O	z	u	w	v	b
SECURITY - PATH	<div>A255</div> <div>Full path and file name of an executable file (for example, DLL for Windows or shared library for UNIX) containing the user security exit which the kernel will load and call. Example:</div> <div>SECURITY - PATH=usersec.dll</div> <div>This assumes the DLL is in the default path. Or:</div> <div>SECURITY - PATH=c:\brokerexit\yoursecu.dll</div> <div>If the path name contains spaces, enclose it in quotation marks. Example:</div>	O	z	u	w		b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
	SECURITY-PATH="c:\Software AG\broker exit\yoursecu.dll"						
	Note: This attribute is used only when implementing a user-written security exit.						
SERVER-DEFAULT	n UNLIM	O	z	u	w	v	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	v	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 <i>without</i> 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	v	b
	Number of short buffers to be allocated for each 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. 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	v	b
	Create a storage report about broker memory usage. NO Do not create the storage report. YES Create the storage report.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	See <i>Storage Report</i> .						
STORE	OFF BROKER	O	z	u	w	v	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 <i>Flushing Trace Data to a GDG Data Set</i> under <i>Tracing EntireX Broker</i> . 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: <div>TRACE-DD = "DSNAME=EXX.GDG, DCB=(BLKSIZE=1210,DSORG=PS,LRECL=121,RECFM=FB), DISP=(NEW,CATLG,CATLG), SPACE=(CYL,(100,10)), STORCLAS=SMS"</div> 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 nK nM nG	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						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
	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 <i>Trace File Handling</i> in the UNIX and Windows Administration documentation.						
TRACE - LEVEL	0 - 4	O	z	u	w	v	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			v	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 UNIX or Windows. Examples: TRANSPORT=NET specifies that only the Entire Net-Work transport method will be supported by the broker. TRANSPORT=TCP-NET specifies that both the TCP/IP and Net-Work transport methods will be supported by the broker.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
	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 <i>Deferred Tracing</i> in the platform-specific Broker 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	<u>1D</u> nS nM nH nD	O	z	u	w	v	b
	Defines the default lifetime for units of work for the service. nS Number of seconds the UOW can exist (max. 2147483647). nM Number of minutes the UOW can exist (max. 35791394). nH Number of hours the UOW can exist (max. 596523). nD 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 <i>Timeout Considerations for EntireX Broker</i> .						
UOW-MSGS	See MAX-MESSAGES-IN-UOW.						
UOW-STATUS-LIFETIME	<u>no value</u> n[S] nM nH nD	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>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 <i>status</i> information will be the same as the lifetime of the UOW itself.</p> <p><i>nS</i> Number of seconds the UOW status exists longer than the UOW itself (max. 2147483647).</p> <p><i>nM</i> Number of minutes (max. 35791394).</p> <p><i>nH</i> Number of hours (max. 596523).</p> <p><i>nD</i> Number of days (max. 24855).</p> <p>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.</p> <p>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.</p>						
UWSTAT-LIFETIME	Alias for UOW-STATUS-LIFETIME.						
UWSTATP	<u>Q</u> <i>n</i>	O	z	u	w	v	b
	<p>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.</p> <p>0 The status is not persistent.</p> <p>1 - 254 Multiplied by the value of UOW-DATA-LIFETIME to determine how long a persistent status will be retained.</p> <p>Note: This attribute has not been supported since EntireX version 7.3. Use UOW-STATUS-LIFETIME instead.</p>						
UWTIME	Alias for UOW-DATA-LIFETIME.						
WAIT-FOR-ACTIVE-PSTORE	<u>NO</u> YES	O	z	u	w	v	b
	Determines whether broker should wait for the Adabas Persistent Store to become active, or until c-tree PSTORE files become available.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>NO If broker should start with a PSTORE - TYPE=ADABAS and the database is not active or is not accessible, broker will stop.</p> <p>If broker should start with a PSTORE - TYPE=CTREE and the c-tree files are still in use, broker will stop.</p> <p>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 with the PSTORE. Broker will reject any user requests until it is able to contact the Adabas database.</p> <p>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.</p>						
WORKER-MAX	<u>32</u> <i>n</i> (min. 1, max. 32)	O	z	u	w		b
	Maximum number of worker tasks the broker can use.						
WORKER-MIN	<u>1</u> <i>n</i> (min. 1, max. 32)	O	z	u	w		b
	Minimum number of worker tasks the broker can use.						
WORKER-NONACT	<u>70S</u> <i>n</i> <i>nS</i> <i>nM</i> <i>nH</i>	O	z	u	w		b
	<p>Non-activity time to elapse before a worker tasks is stopped.</p> <p><i>n</i> Same as <i>nS</i>.</p> <p><i>nS</i> Non-activity time in seconds (default 70, max. 2147483647).</p> <p><i>nM</i> Non-activity time in in minutes (max. 35791394).</p> <p><i>nH</i> Non-activity time in hours (max. 596523).</p> <p>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.</p>						
WORKER-QUEUE-DEPTH	<u>1</u> <i>n</i> (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	<i>internal-value</i> <i>n</i>	O	z	u	w		b
	<i>n</i> Delay is extended by <i>n</i> seconds.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>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.</p> <p>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.</p>						

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* under *Broker Attributes* and [Service Update Modes](#) below the table.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
APPLICATION-MONITORING or APPMON	YES <u>NO</u>	O	z	u	w	v	
	YES Enable application monitoring for the specified services. NO Disable application monitoring for the specified services. See <i>Application Monitoring</i> .						
APPLICATION-MONITORING-NAME or APPMON-NAME	A100	O	z	u	w	v	
	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 <i>Application Monitoring</i> .						
CLASS	A32 (case-sensitive)	R	z	u	w	v	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. Classes starting with any of the following are reserved for use by Software AG and should not be used in customer-written applications: 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	<u>N</u> Y	O	z				b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>Determines whether this service is subject to RPC authorization checking.</p> <p>N No RPC authorization checking is performed.</p> <p>Y RPC library and program name are appended to the authorization check performed by EntireX Security. Specify "YES" only to RPC-supported services.</p> <p>To allow conformity with Natural Security, the CLIENT - RPC - AUTHORIZATION parameter can optionally be defined with a prefix character as follows: CLIENT - RPC - AUTHORIZATION= (YES, <prefix-character>).</p>						
CONV - LIMIT	<u>UNLIM</u> <i>n</i>	O	z	u	w	v	b
	<p>Allocates a number of conversations especially for this service.</p> <p>UNLIM The number of conversations is restricted only by the number of conversations globally available. Precludes the use of NUM - CONVERSATION= AUTO in the Broker section of the attribute file.</p> <p><i>n</i> Number of conversations.</p> <p>A value of 0 (zero) is invalid.</p> <p>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 the CONV - LIMIT attribute must be suppressed entirely for the service so that the default (CONV - DEFAULT) becomes active.</p>						
CONV - NONACT	<u>5M</u> <i>n</i> <i>nS</i> <i>nM</i> <i>nH</i>	R	z	u	w	v	b
	<p>Non-activity time for connections.</p> <p><i>n</i> Same as <i>nS</i>.</p> <p><i>nS</i> Non-activity time in seconds (max. 2147483647).</p> <p><i>nM</i> Non-activity time in minutes (max. 35791394).</p> <p><i>nH</i> Non-activity time in hours (max. 596523).</p> <p>A value of 0 (zero) is invalid. If a connection is not used for the specified time, that is, a server 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.</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
CONVERSION	Format: A255 (SAGTCHA [, TRACE =n] [, <i>OPTION</i> =s] SAGTRPC [, TRACE =n] [, <i>OPTION</i> =s] name [, TRACE =n] NO)	O	z	u	w	v	b
<p>Defines ICU conversion or SAGTRPC user exit for character conversion. See <i>Internationalization with EntireX</i>.</p> <p>SAGTCHA ⁽¹⁾ Conversion using ICU Conversion for <i>ACI-based Programming</i>.</p> <p>SAGTRPC ⁽²⁾ Conversion using ICU Conversion for <i>RPC-based Components and Reliable RPC</i>.</p> <p><name> ⁽³⁾ Name of the SAGTRPC user exit for <i>RPC-based components and Reliable RPC</i>. See also <i>Configuring SAGTRPC User Exits</i> under <i>Configuring Broker for Internationalization</i> in the platform-specific Administration documentation and <i>Writing SAGTRPC User Exits</i> under <i>Configuring Broker for Internationalization</i> in the platform-specific Administration documentation.</p> <p>NO If conversion is not to be used, either omit the CONVERSION attribute or specify CONVERSION=NO, for example for binary payload.</p> <p>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.</p> <p>Note:</p> <ol style="list-style-type: none"> See also <i>Configuring ICU Conversion</i> under <i>Configuring Broker for Internationalization</i> in the platform-specific Administration documentation. SAGTRPC is not supported on BS2000. For conversion with single-byte code pages, use SAGTCHA on BS2000 for <i>RPC-based Components and Reliable RPC</i>. 							

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	3. SAGTRPC user exit is not supported on z/VSE and BS2000.						
	TRACE						
	If tracing is switched on, the trace output is written to the broker log file:						
	0 No tracing						
	1 Trace level 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. Please note that if <i>OPTION Values for Conversion</i> are set, errors are ignored.						
	2 Trace level ADVANCED Tracing of incoming, outgoing parameters and the payload.						
	3 Trace level SUPPORT This trace level is for support diagnostics and should only be switched on when requested by Software AG support.						
	OPTION						
	See table of possible values under <i>OPTION Values for Conversion</i> .						
DEFERRED	<u>NO</u> YES	O	z	u	w	v	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	<u>YES</u> NO	O	z	u	w	v	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. NO A new conversation is always assigned to the first server in the queue.						
LONG-BUFFER-LIMIT	<u>UNLIM</u> n	O	z	u	w	v	b
	Allocates a number of long message buffers for the service.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>UNLIM The number of long message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM- LONG- BUFFER= AUTO in the Broker section of the attribute file.</p> <p><i>n</i> Number of long message buffers.</p> <p>A value of 0 (zero) is invalid. If NUM- LONG- BUFFER= AUTO is specified in the Broker section of the attribute file, LONG- BUFFER- LIMIT= UNLIM is not allowed in the service section. A value must be specified or the LONG- BUFFER- LIMIT attribute must be suppressed entirely for the service so that the default (LONG- BUFFER- DEFAULT) becomes active.</p>						
MAX-MESSAGES-IN-UOW	16 <i>n</i>	O	z	u	w	v	b
	Maximum number of messages in a UOW.						
MAX-MESSAGE-LENGTH	2147483647 <i>n</i>	O	z	u	w		b
	Maximum message size that can be sent to a service.						
	This is transport-dependent. The default value represents the highest positive number that can be stored in a four-byte integer.						
MAX-MSG	See MAX-MESSAGE-LENGTH.						
MAX-UOW-MESSAGE-LENGTH	See MAX-MESSAGE-LENGTH.						
MAX-UOWS	0 <i>n</i>	O	z	u	w	v	b
	<p>0 The service does not accept units of work, i.e. it processes only messages that are not part of a UOW. Using zero prevents the sending of UOWs to services that are not intended to process them.</p> <p><i>n</i> Maximum number of UOWs that can be active concurrently for the service. If you do not provide a MAX-UOWS value for the service, it defaults to the MAX-UOWS setting for the broker. If you provide a value that exceeds that of the broker, the service MAX-UOWS is set to the broker's MAX-UOWS value and a warning message is issued.</p> <p>Specify MAX-UOWS=0 for Natural RPC Servers. This restriction will be removed with a later release.</p>						
MUOW	See MAX-UOWS.						
NOTIFY-EOC	NO YES	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>Specifies whether timed-out conversations are to be stored or discarded.</p> <p>NO Discard the EOC notifications if the server is not ready to receive.</p> <p>YES Store the EOC notifications if the server is not ready to receive and then notify the server if possible.</p> <p>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.</p> <p>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.</p>						
NUM-UOW	Alias for MAX-UOWS.						
POSTPONE-ATTEMPTS	<u>Q</u> <i>n</i>	O	z	u	w		
	<p>Defines the number of attempts putting a received unit of work (UOW) due to SYNCPOINT option CANCEL on the postpone queue for later processing.</p> <p>0 All UOWs rejected by the receiver (SYNCPOINT option CANCEL) will be cancelled immediately. Attribute POSTPONE-DELAY is ignored.</p> <p><i>n</i> 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 postpone 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.</p> <p>The default value is 0. See <i>Postponing Units of Work</i>.</p>						
POSTPONE-DELAY	<u>Q</u> <i>n</i> <i>nS</i> <i>nM</i> <i>nH</i>	O	z	u	w		
	<p>The length of time a UOW is kept in status POSTPONED.</p> <p>0 The postpone feature is disabled. Attribute POSTPONE-ATTEMPTS is ignored.</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
	<p><i>nS</i> Number of seconds the UOW stays unreadable in the postpone queue with status POSTPONED (max. 2147483647).</p> <p><i>nM</i> Number of minutes the UOW stays unreadable in the postpone queue with status POSTPONED (max. 35791394).</p> <p><i>nH</i> Number of hours the UOW stays unreadable in the postpone queue with status POSTPONED (max. 596523).</p> <p><i>nD</i> Number of days the UOW stays unreadable in the postpone queue with status POSTPONED (max. 24855).</p> <p>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.</p> <p>Note: By default, the postpone feature is disabled. However, if any value is specified, the minimum delay is 30 seconds. Any value entered that is less than 30 seconds will be increased to this value.</p>						
SERVER	A32 (case-sensitive)	R	z	u	w	v	b
	<p>Part of the name that identifies the service together with the CLASS and SERVICE attributes.</p> <p>CLASS must be specified first, followed immediately by SERVER and SERVICE.</p> <p>Valid characters for server name are letters a-z, A-Z, numbers 0-9, hyphen and underscore. Do not use dollar, percent, period or comma.</p>						
SERVER-DEFAULT	<i>n</i> UNLIM	O	z	u	w	v	b
	<p>Default number of servers that are allowed for every service.</p> <p><i>n</i> Number of servers.</p> <p>UNLIM The number of servers is restricted only by the number of servers globally available. Precludes the use of NUM-SERVER=AUTO.</p> <p>A value of 0 (zero) is invalid.</p> <p>This value can be overridden by specifying a SERVER-LIMIT for the service.</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
SERVER-LIMIT	<i>n</i> UNLIM	O	z	u	w	v	b
<p>Allows a number of servers especially for this service.</p> <p><i>n</i> Number of servers.</p> <p>UNLIM The number of servers is restricted only by the number of servers globally available. Precludes the use of NUM-SERVER=AUTO in the Broker section of the attribute file.</p> <p>A value of 0 (zero) is invalid.</p> <p>If NUM-SERVER=AUTO is specified in the Broker section of the attribute file, SERVER-LIMIT=UNLIM is not allowed in the service section. A value must be specified or the SERVER-LIMIT attribute must be suppressed entirely for the service so that the default (SERVER-DEFAULT) becomes active.</p> <p>Note: UNIX and Windows: This limit also includes any attach server you are using. Make sure you increase the number by one for each attach server you use.</p>							
SERVER-NONACT	<u>5M</u> <i>n</i> <i>nS</i> <i>nM</i> <i>nH</i>	R	z	u	w	v	b
<p>Non-activity time for servers. A server that does not issue a broker request within the specified time limit is treated as inactive and all resources for the server are freed.</p> <p><i>n</i> Same as <i>nS</i>.</p> <p><i>nS</i> Non-activity time in seconds (max. 2147483647).</p> <p><i>nM</i> Non-activity time in minutes (max. 35791394).</p> <p><i>nH</i> Non-activity time in hours (max. 596523).</p> <p>If a server registers multiple services, the highest value of all the services registered is taken as non-activity time for the server.</p>							
SERVICE	A32 (case-sensitive)	R	z	u	w	v	b
<p>Part of the name that identifies the service together with the CLASS and SERVER attributes.</p> <p>CLASS must be specified first, followed immediately by SERVER and SERVICE.</p>							

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
	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	<u>UNLIM</u> <i>n</i>	O	z	u	w	v	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. <i>n</i> 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	<u>OFF</u> <u>BROKER</u>	O	z	u	w	v	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	Format: A255 NO <i><name></i>	O	z	u	w	v	b
	Activates translation user exit for character conversion. NO If translation is not to be used - e.g., for binary payload (broker messages) - either omit the TRANSLATION attribute or specify TRANSLATION=NO. <i><name></i> Name of Translation User Exit. See also <i>Configuring Translation User Exits</i> under <i>Configuring Broker</i> for						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p><i>Internationalization</i> in the platform-specific Administration documentation or <i>Writing Translation User Exits</i> under <i>Configuring Broker for Internationalization</i> in the platform-specific Administration documentation.</p> <p>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.</p>						
UMSG	Alias for MAX-MESSAGES-IN-UOW.						
UOW-DATA-LIFETIME	<u>1D</u> <i>nS</i> <i>nM</i> <i>nH</i> <i>nD</i>	O	z	u	w	v	b
	<p>Defines the default lifetime for units of work for the service.</p> <p><i>nS</i> Number of seconds the UOW can exist (max. 2147483647). <i>nM</i> Number of minutes the UOW can exist (max. 35791394). <i>nH</i> Number of hours the UOW can exist (max. 596523). <i>nD</i> Number of days the UOW can exist (max. 24855).</p> <p>If the unit of work (UOW) is inactive, that 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.</p>						
UOW-MSGs	Alias for MAX-MESSAGES-IN-UOW.						
UOW-STATUS-LIFETIME	<u>no value</u> <i>n[S]</i> <i>nM</i> <i>nH</i> <i>nD</i>	O	z	u	w	v	b
	<p>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 <i>status</i> information will be the same as the lifetime of the UOW itself.</p> <p><i>nS</i> Number of seconds the UOW status exists longer than the UOW itself (max. 2147483647). <i>nM</i> Number of minutes (max. 35791394). <i>nH</i> Number of hours (max. 596523). <i>nD</i> Number of days (max. 24855).</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
	<p>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.</p> <p>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.</p>						
UWSTATP	<u>Q</u> <i>n</i>	O	z	u	w	v	b
	<p>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.</p> <p>0 The status is not persistent.</p> <p>1 - 254 Multiplied by the value of UOW-DATA-LIFETIME to determine how long a persistent status will be retained.</p> <p>Note: This attribute has not been supported since EntireX version 7.3. Use UOW-STATUS-LIFETIME instead.</p>						
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 can not 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 codepage-dependent default replacement character.	yes	yes	No message.	No message.
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	UNIX	Windows	z/VSE	BS2000
DEFAULT_ASCII	Any ICU converter name or alias. See also Additional Notes below.	O	z	u	w	v	b
<p>Customize the broker's locale string defaults by assigning the default codepage for EntireX components (client or server). See <i>Broker's Locale String Defaults</i>. This value is used instead of the broker's locale string defaults if</p> <ul style="list-style-type: none"> ■ the calling component does not send a locale string itself, and ■ the calling component is running on an ASCII platform (UNIX, Windows, etc.) <p>Example:</p> <pre>DEFAULTS=CODEPAGE * Broker Locale String Defaults DEFAULT_ASCII=windows-950</pre> <p>For more examples, see <i>Configuring Broker's Locale String Defaults</i> and also Additional Notes below.</p>							
DEFAULT_EBCDIC_IBM	Any ICU converter name or alias	O	z	u	w	v	b
<p>Customize the broker's locale string defaults by assigning the default codepage for EntireX components (client or server). See <i>Broker's Locale String Defaults</i>. This value is used instead of the broker's locale string defaults if</p> <ul style="list-style-type: none"> ■ the calling component does not send a locale string itself and 							

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<ul style="list-style-type: none">■ the calling component is running on an IBM mainframe platform (z/OS, z/VSE etc.) <p>Example:</p> <pre>DEFAULT=CODEPAGE DEFAULT_EBCDIC_IBM=ibm-937</pre> <p>For more examples, see <i>Configuring Broker's Locale String Defaults</i> and also Additional Notes below.</p>						
DEFAULT_EBCDIC_SNI	Any ICU converter name or alias	O	z	u	w	v	b
	<p>Customize the broker's locale string defaults by assigning the default codepage for EntireX components (client or server). See <i>Broker's Locale String Defaults</i>. This value is used instead of the locale string defaults if</p> <ul style="list-style-type: none">■ the calling component does not send a locale string itself, and■ the calling component is running on a Fujitsu EBCDIC mainframe platform (BS2000) <p>Example:</p> <pre>DEFAULT=CODEPAGE DEFAULT_EBCDIC_SNI= bs2000-edf03drv</pre> <p>For more examples, see <i>Configuring Broker's Locale String Defaults</i> and also Additional Notes below.</p>						
locale-string	Any ICU converter name or alias. See also Additional Notes below.	O	z	u	w	v	
	<p>Customize the mapping of locale strings to codepages and bypass the broker's locale string processing mechanism. See <i>Broker's Locale String Processing</i>. This is useful:</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<ul style="list-style-type: none">■ if the broker's locale string processing fails - i.e. 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 <i>Building and Installing ICU Custom Converters</i> in the platform-specific Administration documentation. <p>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 <i>Broker's Built-in Locale String Mapping</i>. Example:</p> <pre>DEFAULTS=CODEPAGE * Broker Locale String Codepage Assignments ASCII=ISO8859 EUC_JP_LINUX=ibm-33722_P12A-1999 * Customer-written ICU converters CP1140=myebcdic CP0819=myascii</pre> <p>For more examples, see <i>Bypassing Broker's Built-in Locale String Mapping</i> and also Additional Notes below.</p>						

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 is 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 UNIX and Windows.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
ADASVC	<i>nnn</i>	R	<i>z</i>			<i>v</i>	
<p>Sets the Adabas SVC number for EntireX Broker access.</p> <p>The Adabas SVC is used to perform various internal functions, including communication between the caller program and EntireX Broker.</p> <p>Not supported on BS2000.</p>							
EXTENDED-ACB-SUPPORT	<u>NO</u> YES	O	<i>z</i>			<i>v</i>	<i>b</i>
<p>Determines whether extended features of Adabas version 8 (or above) are supported.</p> <p>NO No features of Adabas version 8 or above will be used.</p> <p>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.</p>							
FORCE	<u>NO</u> YES	O	<i>z</i>			<i>v</i>	<i>b</i>
<p>Determines whether DBID table entries can be overwritten.</p> <p>NO Overwrite of DBID table entries not permitted.</p> <p>YES Overwrite of DBID table entries permitted. This is required when the DBID table entry is not deleted after abnormal termination.</p> <p>Caution: Overwriting an existing entry prevents any further communication with the overwritten node. Use <code>FORCE=YES</code> only if you are absolutely sure that no target node with that DBID is active.</p>							

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
IDTNAME	FORMAT: A8 <i>idtname</i> 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. 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 <i>n</i>	O	z			v	b
	<p>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 <i>Adabas Operations Manual</i>).</p> <p>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.</p>						
LOCAL	NO YES	O	z			v	b
	<p>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.</p> <p>NO DBID is <i>global</i> and can be accessed from remote nodes via Entire Net-Work. YES DBID is <i>local</i> and cannot be accessed from remote nodes via Entire Net-Work.</p>						
MAX-MESSAGE-LENGTH	2147483647 <i>n</i>	O	z	u	w	v	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 <i>n</i>	O	z			v	b
	<p>The number of attached buffers to be used (max. 524287).</p> <p>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.</p> <p>The following formula can be used to calculate the value for NABS:</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	NABS = NCQE *IUBL / 4096.						
NCQE	<u>10</u> <i>n</i>	O	z			v	b
	<p>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.</p> <p>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:</p>						
NODE	1-65534	R	z			v	b
	<p>Defines the unique DBID for EntireX Broker.</p> <p>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.</p>						
TIME	<u>30</u> <i>n</i>	O	z			v	b
	<p>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.</p>						
TRACE - LEVEL	<u>0</u> - 4	O	z			v	b
	<p>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.</p> <p>0 No tracing. Default value.</p> <p>1 Display invalid Adabas commands.</p> <p>2 All of trace level 1, plus errors if request entries could not be allocated.</p> <p>3 All of trace level 2, plus all routines executed.</p> <p>4 All of trace level 3, plus function arguments and return values.</p> <p>Trace levels 2, 3 and 4 should be used only when requested by Software AG support.</p> <p>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.</p>						

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	UNIX	Windows	z/VSE	BS2000
ACCESS-SECURITY-SERVER	<u>NO</u> YES	O					b
	<p>Determines where authentication is checked.</p> <p>NO Authentication is checked in the broker tasks. This requires broker to be running under TSOS in order to execute privileged security checks.</p> <p>YES Authentication is checked in the EntireX Broker Security Server for BS2000. This does not require broker to be running under TSOS. See <i>EntireX Broker Security Server for BS2000</i>.</p>						
APPLICATION-NAME	A8	O	z				
	<p>Specifies the name of the application to be checked if <code>FACILITY-CHECK=YES</code> is defined. In RACF, for example, an application "BROKER" with read permission for user "DOE" is defined with following commands:</p> <pre>RDEFINE APPL BROKER UACC(NONE) PERMIT BROKER CLASS(APPL) ID(DOE) ACCESS(READ) SETROPTS CLASSACT(APPL)</pre> <p>See attribute <code>FACILITY-CHECK</code> for more information.</p>						
AUTHORIZATION-DEFAULT	<u>YES</u> NO	O		u	w		
	<p>Determines whether access is granted to a specified service if the specified could not be found listed in the repository of authorization rules or in section <code>DEFAULTS=AUTHORIZATION-RULES</code> of the attribute file.</p> <p>YES Grant access.</p> <p>NO Deny access.</p> <p>Applies only when using EntireX Security under UNIX 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.</p> <p>See also <i>Authorization Rules</i>.</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
CHECK-IP-ADDRESS	YES <u>NO</u>	O	z				
	Determines whether the TCP/IP address of the caller is subject to a resource check.						
ERRTXT-MODULE	<u>NA2MSG0</u> NA2MSG1 NA2MSG2 <i>ModuleName</i>	O	z				
	Specifies the name of the security error text module. Default is "NA2MSG0", English messages. For instructions on how to customize messages, see <i>Build Language-specific Messages (Optional)</i> under <i>Installing EntireX Security under z/OS</i> .						
FACILITY-CHECK	<u>NO</u> 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	<u>NO</u> YES	O	z	u	w		b
	Determines whether the value of the ACI field SECURITY-TOKEN is verified on each call.						
INCLUDE-CLASS	<u>YES</u> NO	O	z				
	Determines whether the class name is included in the resource check.						
INCLUDE-NAME	<u>YES</u> NO	O	z				
	Determines whether the server name is included in the resource check.						
INCLUDE-SERVICE	<u>YES</u> NO	O	z				
	Determines whether the service name is included in the resource check.						
LDAP-AUTHENTICATION-URL	<i>ldapUrl</i>	O		u	w		
	Authentication is performed against the LDAP repository specified under <i>ldapUrl</i> . ■ TCP Specify repository URL:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	BS2000
	<div>LDAP-AUTHENTICATION-URL="ldap://HostName[:PortNumber]"</div> <div>■ SSL/TLS Specify repository URL with ldaps:</div> <div>LDAP-AUTHENTICATION-URL="ldaps://HostName[:PortNumber]"</div> <div>If no port number is specified, the default is the standard LDAP port number 389 for TCP transport. Examples for TCP and SSL/TLS:</div> <div>LDAP-AUTHENTICATION-URL="ldap://myhost.mydomain.com" LDAP-AUTHENTICATION-URL="ldaps://myhost.mydomain.com:636"</div>						
LDAP-AUTHORIZATION-URL	<i>ldapUrl</i>	O		u	w		
	<div>Authorization is performed against the LDAP repository specified under <i>ldapUrl</i>.</div> <div>■ TCP Specify repository URL:</div> <div>LDAP-AUTHORIZATION-URL="ldap://HostName[:PortNumber]"</div> <div>If no port number is specified, the default is the standard LDAP port number 389 for TCP transport. Example for TCP:</div> <div>LDAP-AUTHORIZATION-URL="ldap://myhost.mydomain.com:389"</div> <div>This attribute replaces the parameters <i>host</i>, <i>port</i> and <i>protocol</i> in the <i>xds.ini</i> file of EntireX version 9.10 and below.</div>						
LDAP-AUTH-DN	<i>authDN</i>	O		u	w		
	<div>For authenticated access to the LDAP server. Specifies the DN of the user. Default value:</div> <div>cn=admin,dc=software-ag,dc=de</div> <div>This attribute replaces parameter <i>authDN</i> in the <i>xds.ini</i> file of EntireX version 9.10 and below.</div>						
LDAP-AUTH-PASSWD-ENCRYPTED	<i>authPass</i>	O		u	w		
	<div>For authenticated access to the LDAP server. Specifies the encrypted value of the user password. Use program <i>etbnattr</i> to get the encrypted password:</div>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<code>etbnattr -x clear_text_password -echo_password_only</code> This writes the encrypted password to standard output. This attribute replaces parameter <code>authPass</code> in the <code>xds.ini</code> file of EntireX version 9.10 and below.						
LDAP - AUTHORIZATION - RULE	A32	O		u	w		
	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 UNIX or Windows and <code>SECURITY - SYSTEM=lldapUrl</code> . Authorization rules can be stored in an LDAP repository. When an authorization call occurs, EntireX Security uses the values of this parameter and <code>AUTHORIZATION - DEFAULT</code> to perform an access check for a particular broker instance against an (authenticated) user ID and list of rules. See also <i>Authorization Rules</i> .						
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: <code>dc=software-ag,dc=de</code> This attribute replaces parameter <code>baseDN</code> in the <code>xds.ini</code> file of EntireX version 9.10 and below.						
LDAP - PERSON - BASE - BINDDN	lldapDn	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: <code>LDAP - PERSON - BASE - BINDDN="cn=users,dc=mydomain,dc=com"</code>						
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 <code>domainName\userId</code> .						
LDAP - SASL -	NO YES	O			w		

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
AUTHENTICATION	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	<u>cn</u> <i>uidFieldName</i>	O		u	w		
	Used with LDAP authentication to specify the first field name of a user in the Distinguished Name, for example: LDAP-USERID-FIELD= <i>uid</i>						
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	<u>NO</u> YES	O	z			v	
	Determines whether the password and new password are converted to uppercase before verification.						
PRODUCT	<u>RACF</u> 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: ACF2 Security system ACF2 is installed. RACF Security system RACF is installed. Default. TOP-SECRET Security system TOP-SECRET is installed. The default value is used if an incorrect or no value is specified.						
PROPAGATE-TRUSTED-USERID	<u>YES</u> 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 <code>CLIENT-USERID</code> .						
SAF-CLASS	<u>NBKSAG</u> <i>SAFClassName</i>	O	z				
	Specifies the name of the SAF class/type used to hold the EntireX-related resource profiles.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
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	v	b
	<p>Specifies the mode of operation.</p> <p>AUTHORIZATION Authorization and authentication (not under BS2000 or z/VSE). AUTHENTICATION Authentication.</p> <p>Note: In version 8.0, the default value for this parameter was "AUTHORIZATION".</p>						
SECURITY-NODE	YES name	O	z				
	<p>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.</p> <p>YES This causes the broker ID to be used as a prefix for all authorization checks. name This causes the actual text (maximum 8 characters) to be prefixed onto all authorization checks.</p> <p>Note: By <i>not</i> setting this parameter, no prefix is added to the resource check (the default behavior).</p>						
SECURITY-SYSTEM	OS LDAP	O	z	u	w		b
	<p>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.</p> <p>LDAP Authentication and authorization are performed against the LDAP repository specified under LDAP-AUTHENTICATION-URL and LDAP-AUTHORIZATION-URL.</p>						
TRACE-LEVEL	0 - 4	O	z	u	w	v	b
	<p>Trace level for EntireX Security. It overrides the global value of trace level in the attribute file.</p> <p>0 No tracing. Default value. 1 Log security violations and access denied/permited.</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>2 All of trace level 1, plus internal errors.</p> <p>3 All of trace level 2, plus function entered/exit messages with argument values and some progress messages.</p> <p>4 All of trace level 3, plus some selected data areas for problem analysis.</p> <p>Trace levels 2, 3 and 4 should be used only when requested by Software AG support.</p> <p>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.</p> <p>Note: Setting this value also affects tracing for authorization rules.</p>						
TRUSTED-USERID	<u>YES</u> NO	O	z				
	Activates the trusted user ID mechanism for broker requests arriving over the local Adabas IPC mechanism.						
USERID-TO-UPPER-CASE	<u>NO</u> YES	O	z			v	
	Determines whether user ID is converted to uppercase before verification.						
UNIVERSAL	<u>NO</u> YES	O	z				
	Determines whether access to undefined resource profiles is allowed.						
WARN-MODE	<u>NO</u> 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	UNIX	Windows	z/SE	BS2000
CONNECTION-NONACT	<i>n</i> <i>nS</i> <i>nM</i> <i>nH</i>	O	z	u	w	v	b
<p>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.</p> <p><i>n</i> Same as <i>nS</i>. <i>nS</i> Non-activity time in seconds (min. 600, max. 2147483647). <i>nM</i> Non-activity time in minutes (min. 10, max. 35791394). <i>nH</i> Non-activity time in hours (max. 596523).</p> <p>If not specified, the connection non-activity test is disabled. On the stub side, non-activity can be set with the environment variable <code>ETB_NONACT</code>. See <i>Limiting the TCP/IP Connection Lifetime</i> in the platform-specific <i>Stub Administration</i> sections of the EntireX documentation.</p>							
HOST	<u>0.0.0.0</u> <i>HostName</i> <i>IP</i> <i>address</i>	O	z	u	w	v	b
<p>The address of the network interface on which broker will listen for connection requests.</p> <p>If <code>HOST</code> is not specified, broker will listen on any attached interface adapter of the system (or stack).</p> <p>A maximum of five <code>HOST/PORT</code> pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.</p>							
MAX-MESSAGE-LENGTH	<u>2147483647</u> <i>n</i>	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/SE	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	v	b
	<p>The TCP/IP port number on which the broker will listen for connection requests.</p> <p>If not specified, the broker will attempt to find its TCP/IP port number from the TCP/IP services file, using <code>getservbyname</code>. If it cannot find the number here, the default value of 1971 is used.</p> <p>A maximum of five <code>HOST/PORT</code> pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.</p>						
RESTART	<u>YES</u> NO	O	z	u	w	v	b
	<p>YES The broker kernel will attempt to restart the TCP/IP communicator.</p> <p>NO The broker kernel will not try to restart the TCP/IP communicator.</p> <p>This setting applies to all TCP/IP communicators.</p>						
RETRY-LIMIT	<u>20</u> <i>n</i> UNLIM	O	z	u	w	v	b
	Maximum number of attempts to restart the TCP/IP communicator. This setting applies to all TCP/IP communicators.						
RETRY-TIME	<u>3M</u> <i>n</i> <i>n</i> S <i>n</i> M <i>n</i> H	O	z	u	w	v	b
	<p>Wait time between stopping the TCP/IP communicator due to an unrecoverable error and the next attempt to restart it.</p> <p><i>n</i> Same as <i>n</i> S.</p> <p><i>n</i> S Wait time in seconds (max. 2147483647).</p> <p><i>n</i> M Wait time in minutes (max. 35791394).</p> <p><i>n</i> H Wait time in hours (max. 596523).</p> <p>Minimum wait time is 1S.</p> <p>This setting applies to all TCP/IP communicators.</p>						
REUSE-ADDRESS	<u>YES</u> NO	O	z	u		v	b
	YES <u>NO</u>	O			w		

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>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).</p> <p>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.</p> <p>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.</p>						
STACK - NAME	StackName	O	z				
	<p>Name of the TCP/IP stack that the broker is using.</p> <p>If not specified, broker will connect to the default TCP/IP stack running on the machine.</p>						
TRACE - LEVEL	0 - 4	O	z	u	w	v	b
	<p>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.</p> <p>0 No tracing. Default value.</p> <p>1 Display IP address of incoming request, display error number of outgoing error responses.</p> <p>2 All of trace level 1, plus errors if request entries could not be allocated.</p> <p>3 All of trace level 2, plus all routines executed.</p> <p>4 All of trace level 3, plus function arguments and return values.</p> <p>Trace levels 2, 3 and 4 should be used only when requested by Software AG support.</p> <p>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.</p>						

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, BS2000, z/VSE.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
MAXSIZE	<i>n</i> <i>nM</i> <i>nG</i>	O		u	w		
	<p>Defines the maximum size of c-tree data files. Broker allocates one data file for control data and another data file for message data:</p> <p><i>n</i> Maximum size in MB. <i>nM</i> Maximum size in MB. <i>nG</i> Maximum size in GB.</p>						
PAGESIZE	<i>n</i> <i>nK</i>	O		u	w		
	<p>Determines how many bytes are available in each c-tree node. <code>PSTORE COLD start</code> is required after changing this value.</p> <p><i>n</i> Same as <i>nK</i> <i>nK</i> PAGESIZE in KB.</p> <p>The default and minimum value is 8 KB.</p> <p>If PSD Reason Code = 527 is returned during UOW write processing, increase the PAGESIZE value and restart broker with <code>PSTORE=COLD</code>, or migrate the existing PSTORE to a new PSTORE with an increased PAGESIZE value. See <i>Migrating the Persistent Store</i> and define the increased PAGESIZE value for the load broker.</p>						
PATH	A255	O		u	w		
	Path name of the target directory for c-tree index and data files.						
SYNCIO	<u>NO</u> YES	O		u	w		
	<p>Controls the open mode of the c-tree transaction log.</p> <p>NO c-tree transaction log is not opened in synchronous mode. Default.</p> <p>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 <i>c-tree Database as Persistent Store</i> in the UNIX and Windows Administration documentation.</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
TRACE - LEVEL	0 - 4	O		u	w		
<p>Trace level for c-tree persistent store. It overrides the global value of trace level in the attribute file.</p> <p>0 No tracing. Default value.</p> <p>1 Log memory allocation failures and errors during close of files.</p> <p>2 n/a</p> <p>3 All of trace level 1, plus UOWID in use for the various ctree requests and function entered/exit messages.</p> <p>4 All of trace level 3, plus returned function values.</p> <p>Trace levels 2, 3 and 4 should be used only when requested by Software AG support.</p> <p>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.</p>							

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 and Certificates with EntireX*.

The SSL-specific attribute section 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.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
CIPHER-SUITE	<i>string</i>	O	z	u	w		b
<p>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 <i>cipher suite</i>, 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.</p> <p>Under UNIX, Windows and BS2000, the OpenSSL implementation is used; under z/OS it is GSK.</p> <p>The SSL protocol is obsolete and should no longer be used for secure operations. The TLS protocol is the successor of SSL and is readily available in OpenSSL and GSK. The following examples show how to configure strong encryption:</p> <p>■ OpenSSL This example uses FIPS-approved algorithms, but without ADH, MD5 or other low or export-grade implementations. It requires authentication and encryption, that is, do not use NULL ciphers:</p> <pre>CIPHER-SUITE=FIPS:!ADH:!LOW:!EXP:!MD5:!aNULL:!eNULL:@STRENGTH</pre> <p>Default configuration:</p> <pre>CIPHER-SUITE=TLSv1:!ADH:!LOW:!EXP: !MD5:@STRENGTH</pre> <p>See http://www.openssl.org/docs/apps/ciphers.html</p>							

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<div>■ GSK</div> <div>Default configuration:</div> <div>CIPHER-SUITE=35363738392F303132330A1613100D</div> <div>This list of cipher suites starts with a strong '256-bit AES encryption with SHA-1 message authentication and RSA key exchange' (35) and ends with a relatively weak '168-bit Triple DES encryption with SHA-1 message authentication and fixed Diffie-Hellman key exchange signed with a DSA certificate' (0D).</div> <div>See IBM documentation: z/OS V2R1.0 System SSL Programming, SC14-7495-00, Appendix C: Cipher Suite Definitions.</div>						
CONNECTION-NONACT	<i>n</i> <i>nS</i> <i>nM</i> <i>nH</i>	O	z	u	w		b
	<div>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.</div> <div><i>n</i> Same as <i>nS</i>.</div> <div><i>nS</i> Non-activity time in seconds (min. 600, max. 2147483647).</div> <div><i>nM</i> Non-activity time in minutes (min. 10, max. 35791394).</div> <div><i>nH</i> Non-activity time in hours (max. 596523).</div> <div>If not specified, the connection non-activity test is disabled.</div>						
HOST	<i>hostname</i>	O	z	u	w		b
	<div>The address of the network interface on which broker will listen for connection requests.</div> <div>If HOST is not specified, broker will listen on any attached interface adapter of the system (or stack).</div> <div>A maximum of five HOST/PORT pairs can be specified to start multiple instances of EntireX Broker's TCP/IP transport communicator.</div>						
KEY-LABEL	<i>name</i>	O	z				
	<div>The label of the key in the RACF keyring that is used to authenticate the broker kernel (see also TRUST-STORE parameter).</div> <div>Example: "ETBCERT".</div>						
KEY-FILE	<i>file name</i>	R		u	w		b
	<div>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 <i>SSL/TLS Sample Certificates Delivered with EntireX</i>.</div>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Example for UNIX and Windows: <i>MyAppKey.pem</i> . Note: EntireX Broker does not support Java certificates (keystore files of type .jks).						
KEY - PASSWD	<i>password</i> (A32)	R		u	w		b
	Password used to protect the private key. Unlocks the KEY - FILE, for example <i>MyAppKey.pem</i> . Deprecated. See KEY - PASSWD - ENCRYPTED below.						
KEY - PASSWD - ENCRYPTED	<i>encrypted value</i> (A64)	R		u	w		b
	Password used to protect the private key. Unlocks the KEY - FILE, for example <i>MyAppKey.pem</i> . This attribute replaces KEY - PASSWD to avoid a clear-text password as attribute value. If KEY - PASSWD and KEY - PASSWD - ENCRYPTED are both supplied, KEY - PASSWD - ENCRYPTED takes precedence.						
KEY - STORE	<i>file name</i>	R		u	w		b
	SSL certificate; may contain the private key. For test purposes, EntireX delivers certificates for use on various platforms. See <i>SSL/TLS Sample Certificates Delivered with EntireX</i> . Example for UNIX and Windows: <i>ExxAppCert.pem</i> . Note: EntireX Broker does not support Java certificates (keystore files of type .jks).						
MAX - MESSAGE - LENGTH	<u>2147483647</u> <i>n</i>	O	z	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	<i>1025 - 65535</i>	O	z	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 example attribute file. If the port number is not specified, the broker will use the default value of 1958.						
RESTART	<u>YES</u> NO	O	z	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	<u>20</u> <i>n</i> UNLIM	O	z	u	w		b
	Maximum number of attempts to restart the SSL communicator.						
RETRY - TIME	<u>3M</u> <i>n</i> <i>nS</i> <i>nH</i>	O	z	u	w		b
	Wait time between suspending SSL communication due to unrecoverable error and the next attempt to restart it.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p><i>n</i> Same as <i>nS</i>.</p> <p><i>nS</i> Wait time in seconds (max.2147483647).</p> <p><i>nM</i> Wait time in minutes (max. 35791394).</p> <p><i>nH</i> Wait time in hours (max. 596523).</p> <p>Minimum: 1S</p>						
REUSE-ADDRESS	<u>YES</u> NO	O	z	u	w		b
	<p>YES The SSL port assigned to the broker can be taken over and assigned to other applications (this is the default value).</p> <p>NO The SSL port assigned to the broker cannot be taken over and assigned to other applications.</p> <p>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.</p>						
STACK-NAME	<i>name</i>	O	z	u	w		
	<p>Name of the TCP/IP stack that the broker is using.</p> <p>If not specified, broker will connect to the default TCP/IP stack running on the machine.</p>						
TRACE-LEVEL	<u>0</u> - 4	O	z	u	w		b
	<p>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.</p> <p>0 No tracing. Default value.</p> <p>1 Display IP address of incoming request, display error number of outgoing error responses.</p> <p>2 All of trace level 1, plus errors if request entries could not be allocated.</p> <p>3 All of trace level 2, plus all routines executed.</p> <p>4 All of trace level 3, plus function arguments and return values.</p> <p>Trace levels 2, 3 and 4 should be used only when requested by Software AG support.</p> <p>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.</p>						
TRUST-STORE	<i>file name</i> <i>keyring</i>	R	z	u	w		b
	Location of the store containing certificates of trust Certificate Authorities (or CAs).						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<div>z/OS</div> <div>Specify the RACF keyring using the following format: [<i>USER-ID</i> /] <i>RING-NAME</i>. If no value for <i>USER-ID</i> is provided, the keyring is assumed to be associated with the user ID that the broker kernel is running under.</div> <div>BS2000/Windows/UNIX</div> <div>Specify the file name of the CA certificate store. Examples: <i>EXXCACERT.PEM</i>, <i>C:\Certs\ExxCACert.pem</i></div>						
VERIFY-CLIENT	<u>NO</u> YES	O	z	u	w		b
	<div>YES Additional client certificate required.</div> <div>NO No client certificate required (default).</div>						

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	UNIX	Windows	z/VSE	BS2000
DIV	A511	O	z				
	<p>The VSAM persistent store parameters, enclosed in double quotes (""). The value can span more than one line.</p> <p>Note: Deprecated. This attribute is applicable only if you are supplying the persistent store parameters using <i>Format Parameters</i> of earlier versions. We recommend you use the attributes below that were introduced with EntireX 9.12 instead.</p>						
DATASPACE-NAME	A8	O	z				
	<p>Defines the name of the dataspace that will be used to map the persistent store.</p> <p>Default value is DSPSTORE.</p>						
DATASPACE-PAGES	126-524284	O	z				
	<p>Defines the size of the dataspace used to map the persistent store (size=DATASPACE-PAGES * 4 KB). We recommend using the maximum value.</p> <p>Default value is 2048.</p>						
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				
	<p>Trace level for DIV. It overrides the global value of trace level in the attribute file.</p> <p>0 No tracing. Default value.</p> <p>1 Log selected DIV SAVE calls taking longer than 2 seconds elapsed time.</p>						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	<p>2 n/a</p> <p>3 All of trace level 1, plus UOWID in use for the various DIV requests.</p> <p>4 n/a</p> <p>Trace levels 2, 3 and 4 should be used only when requested by Software AG support.</p> <p>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.</p>						

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	UNIX	Windows	z/VSE	BS2000
BLKSIZE	126-20000	O	z	u	w	v	b
	<p>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 <i>Adabas</i> documentation.</p> <p>For reasons of efficiency, do not specify a <code>BLKSIZE</code> 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.</p> <p>The <code>BLKSIZE</code> parameter applies only for a cold start of broker; subsequently the value of <code>BLKSIZE</code> is taken from the last cold start.</p> <p>Default value is 2000.</p>						
DBID	1 - 32535	R	z	u	w	v	b
	Database ID of Adabas database where the persistent store resides.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
FNR	1 - 32535	R	z	u	w	v	b
	File number of broker persistent store file.						
FORCE-COLD	<u>N</u> Y	O	z	u	w	v	b
	<p>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.</p> <p>Specify Y to allow existing information to be overwritten.</p>						
MAXSCAN	0- <i>n</i>	O	z	u	w	v	b
	<p>Limits display of persistent UOW information in the persistent store through Command and Information Services.</p> <p>Default value is 1000.</p>						
OPENRQ	<u>N</u> Y	O	z	u	w	v	b
	Determines whether driver for Adabas persistent store is to issue an OPEN command to Adabas.						
SVC	200-255	R	z			v	
	Use this parameter to specify the Adabas SVC number to be used by the Adabas persistent store driver.						
TRACE-LEVEL	0 - 4	O	z	u	w	v	b
	<p>Trace level for Adabas persistent store. It overrides the global value of trace level in the attribute file.</p> <p>0 No tracing. Default value.</p> <p>1 Log selected Adabas CB fields (command code, response code, subcode, ISN, additions).</p> <p>2 n/a</p> <p>3 All of trace level 1, plus UOWID in use for the various Adabas requests and function entered/exit messages.</p> <p>4 All of trace level 3, plus more Adabas CB fields for successful requests and returned function values.</p> <p>Trace levels 2, 3 and 4 should be used only when requested by Software AG support.</p> <p>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.</p>						

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 *Application Monitoring*.

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

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	UNIX	Windows	z/VSE	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 <i>Rules Stored in Broker Attribute File</i> .						
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. <i>Wildcard Service Definitions</i> under <i>Broker Attributes</i> 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, `DB ID` 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	112
■ Modes of Requesting the Services	113
■ ETBCMD: Executable Command Requests	115
■ ETBINFO: Returnable Information Requests	120

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*.

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
<i>User-Written Interface</i>	application program	■ INFO ■ USER-INFO ■ CMD ■ PARTICIPANT-SHUTDOWN ■ SECURITY-CMD	■ request structures
<i>Command-line Utilities</i>	ETBINFO utility	■ INFO ■ USER-INFO	■ profile ■ command-line parameters
	ETBCMD utility	■ CMD ■ PARTICIPANT-SHUTDOWN ■ SECURITY-CMD	■ command-line parameters

Applicable operating systems: z/OS, UNIX, Windows and z/VSE.

Description of Services

INFO and 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-SHUTDOWN and 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
ALLOW-NEUOWMSGs	New UOW messages are allowed.	3
CLEAR-CMDLOG-FILTER	Remove the specified command log filter.	5
CONNECT-PSTORE	Connects the persistent store. See <i>Availability of Persistent Store</i> .	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 DYNAMIC-WORKER-MANAGEMENT is enabled again.	7
DISCONNECT-PSTORE	Disconnects the persistent store. See <i>Availability of Persistent Store</i> .	4
ENABLE-ACCOUNTING	Enable accounting.	5
ENABLE-CMDLOG	Enable command logging.	5
ENABLE-DYN-WORKER	Enable the DYNAMIC-WORKER-MANAGEMENT again.	

Command Request		Comment	CIS Interface Version
		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-NEUOWMSGs		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
RESUME		Transport ID: NET Snn Tnn. 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-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.
	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	
		QUIESCE	
	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.

Command Request			Comment	CIS Interface Version
			<p>The following steps will be performed:</p> <ul style="list-style-type: none"> ■ 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. 	
		QUIESCE	<p>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).</p>	
	SERVICE <class/server/service>	Internal services cannot be shut down.		7
		IMMED	<p>Caution: All servers offering this service will be deregistered and logged off. The following steps will be performed:</p> <ul style="list-style-type: none"> ■ Error code 00100050 will be replied to all servers, if they are waiting. ■ All existing conversations will be finished with EOC. ■ Users will be logged off. 	
		QUIESCE	<p>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.</p>	
	PARTICIPANT	IMMED	<p>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 <i>Broker CIS Data Structures</i>.</p> <p>The following steps will be performed:</p>	4

Command Request			Comment	CIS Interface Version
			<ul style="list-style-type: none"> ■ 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. <p>Within EntireX Broker nomenclature, a participant is an application implicitly or explicitly logged on to the Broker as a specific user. A participant could act as client or server.</p>	
		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 Snn Tnn	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 Snn Tnn	Check the current status of the transport communicator.	7
STOP	TRANSPORT	Transport ID: NET Snn Tnn	Stop an active or suspended transport communicator. The transport communicator will shut down. All transport-specific resources will be freed. User requests receive response code 148.	7
SUSPEND	TRANSPORT	Transport ID: NET Snn Tnn	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

Command Request			Comment	CIS Interface Version
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: <i>nnnn</i>	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](#).



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

Using Sample Security Exits for Broker Security

8

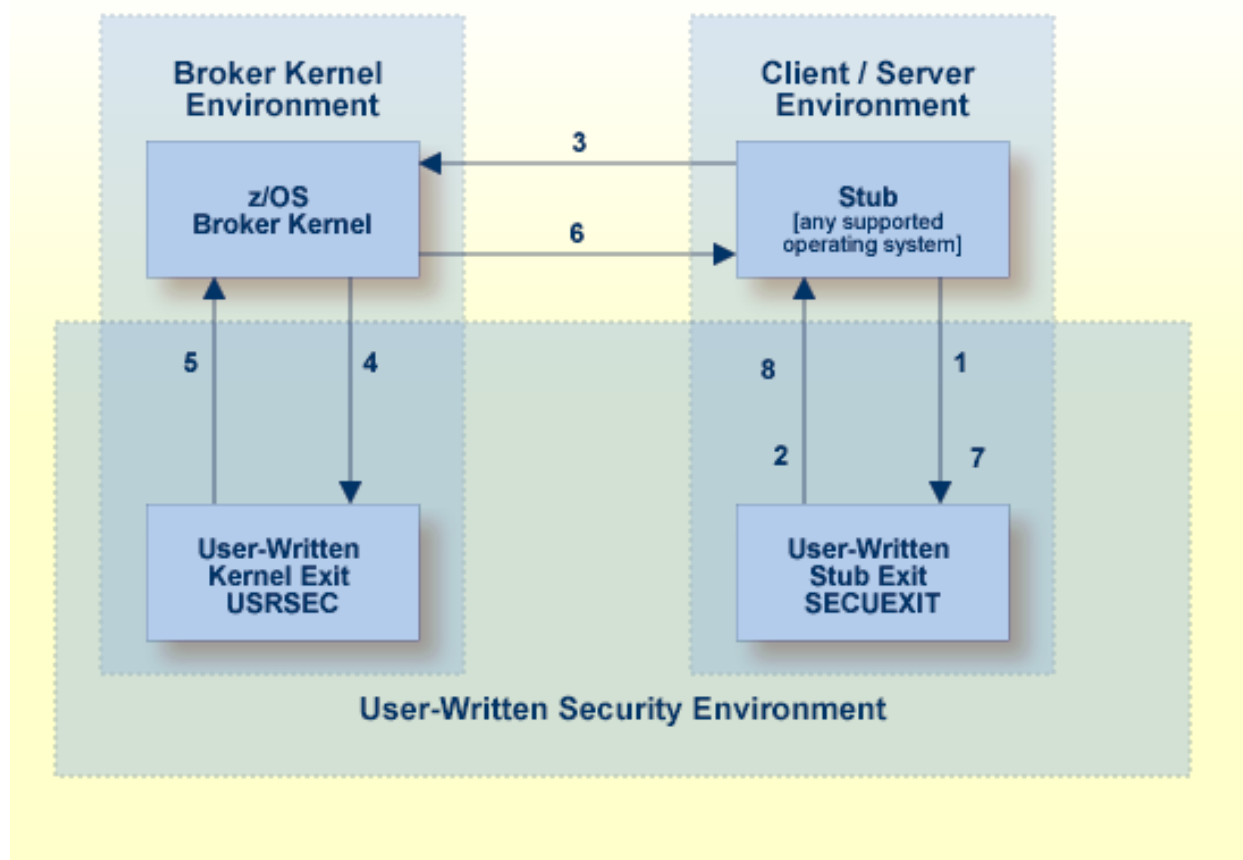
Using Sample Security Exits for Broker Security

■ Overview of Security Data Flow	124
■ Prerequisites for Running EntireX Broker in a Secure Environment	125
■ General Security Recommendations	125
■ Writing Security Exits	127
■ Security-Related Parameters	129
■ Programming Broker Stub Exits	131
■ Required/Recommended Actions in the Exit (depending on Security Type)	135

This page describes implementation issues and how to use sample security exits in EntireX Broker. It assumes you are familiar with EntireX Broker from both an administrative and an application perspective, and with the ACI programming interface in particular. See *Introduction to ACI-based Programming*.

Overview of Security Data Flow

The diagram shows a data flow for sample security exits, with Broker Kernel located, for example, on z/OS. See also *Description of Steps in Data Flow*.



Prerequisites for Running EntireX Broker in a Secure Environment

To run EntireX Broker in a secure environment, the following prerequisites must be met:

- The security system in the EntireX Broker kernel must be activated by setting `SECURITY=YES` in the broker attribute file.
- The security routines must be accessible to the Broker. The method you use to achieve this depends on the operating system where your user-written `USRSEC` is implemented.



Note: EntireX Broker will not start if `SECURITY=YES` is specified but the security routines cannot be activated.

General Security Recommendations

If you run a secure environment, we recommend you performing an explicit `LOGON` with the `AUTOLOGON=NO` definition in the attribute file. All security violations are logged to the EntireX Broker log file.

- [Implementing Security Exits under z/OS](#)
- [Implementing Security Exits under UNIX](#)
- [Implementing Security Exits under Windows](#)

Implementing Security Exits under z/OS

Broker Kernel

➤ To implement the kernel security exit under z/OS

- 1 Write the exits `USRSEC`. The code must always be reentrant and reusable.
- 2 The kernel security exit `USRSEC` is loaded automatically during startup of Broker. Use module and entry name `USRSEC` for this exit. A security module sample source is delivered with the ETB source library.
- 3 Under z/OS, link the exit as reentrant and reusable.
- 4 Ensure that the security exit is accessible in the Broker `STEPLIB`.

Broker Stubs

➤ To implement security exits for Broker stubs under z/OS

- 1 Write the stub security exits `ETBUPRE` and `ETBUEVA`. The code must always be reentrant, except for batch, where the code must be reusable.
- 2 Link these exits `ETBUPRE` and `ETBUEVA` to the stub of the target application. The stub contains weak externals for both entries.

Implementing Security Exits under UNIX

Broker Kernel

➤ To implement the kernel security exit under UNIX

- 1 Write your own *usrsec.c* based on the samples delivered with EntireX.
- 2 Build your own *usrsec.so|l*, using the provided makefiles. (A sample makefile, *makexa*, is provided.).
- 3 Ensure that *usrsec.so|l* is made available to the Broker kernel at execution time. The attribute file parameter `SECURITY-PATH` must be used to specify the location of *usrsec.so|l*.

Broker Stubs

➤ To implement security exits for Broker stubs under UNIX

- 1 Write your own *secuexit.c*, based on the samples delivered with EntireX.
- 2 Build your own *secuexit.so|l*, using the provided makefiles. (A sample makefile, *makexa*, is provided.).
- 3 Ensure that *secuexit.so|l* is made available to the application in the same directory as the Broker stub.

Implementing Security Exits under Windows

Broker Kernel

➤ To implement the kernel security exit under Windows

- 1 Write your own *usrsec.c* based on the samples delivered with EntireX.
- 2 Build your own *usrsec.dll*, using the provided makefiles.

Broker Stubs

➤ To implement security exits for Broker stubs under Windows

- 1 Write your own *secuexit.c*, based on the samples delivered with EntireX.
- 2 Build your own *secuexit.dll*, using the provided makefiles.

Writing Security Exits

This section describes how to write your own security exits. It describes the interfaces, indicates what can be modified and what has to be done within an exit. It also provides some helpful tips.

This section covers the following topics:

- [Requirements](#)
- [Error Checking for Incomplete Security Installation](#)

Requirements

You must provide the following functions:

- The Preparation exit `etbupre()` and the Evaluation exit `etbueva()` for the Broker stub. These two functions are linked statically to the Broker stub routines.
- The Kernel exit `usrsec()` which is loaded by the kernel. This exit is more generic than the other two. It is called with the function that has been performed and a function-dependent Broker ACI control block that provides all the necessary information. This function is loaded dynamically by EntireX Broker during startup. One parameter of the kernel exit is the function that is performed.

The functions map to the exit type is as follows:

Exit Type	Function	Function to perform
Authentication exit	ETB_SEC_LOGON	Checks authentication for the user.
	ETB_SEC_LOGOFF	Release user-specific information if necessary.
	ETB_SEC_NEWPUID	Application call with different physical USER ID.
	ETB_SEC_NEWST	Application call with a different SECURITY TOKEN
Authorization exit	ETB_SEC_SEND	Check whether user is allowed to use the addressed service.
	ETB_SEC_REGISTER	Check whether the user is allowed to offer that service.
Encryption exit	ETB_SEC_ENCRYPT	Encrypt the given data. See Note 1 below.
	ETB_SEC_DECRYPT	Decrypt the given data. See Note 1 below.

**Notes:**

1. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS and Certificates with EntireX*.

In the following text, “encryption” or “authentication” exit refers to the functions listed above.

Error Checking for Incomplete Security Installation

With `ACI_VERSION=4` or above, the security configuration of the caller's stub is checked against the security configuration of the broker kernel. The request will be rejected with the error message 00200379 - API: Inconsistent Security Installation, if security

- is present in the stub and it is not present in the kernel;

or

- is not present in the stub and it is present in the kernel.



Note: If you have written your own security - instead of using *Security Solutions in EntireX* - and it is implemented only on the kernel, you will have to add a dummy security exit to the stub.

Security-Related Parameters

The following security-related parameters are provided. These are fields in the *Broker ACI Fields*:

- USER-ID
- PASSWORD
- SECURITY-TOKEN
- CLIENT-UID
- ERROR-CODE
- ERROR-TEXT
- KERNELSECURITY

USER-ID

The USER ID is defined by the application. It is available in all ACI exits as well as in the kernel exits, except the encryption and decryption exits. Theoretically the preparation exit can be used to retrieve the login name by an operating system specific mechanism. This would allow a user identification without the application being involved. See the description of the `USER-ID` field in the Broker ACI control block.

PASSWORD

The `PASSWORD` is defined by the application. It is available in all ACI and kernel exits except the encryption exit. The `PASSWORD`, if provided by the application in plain text, should be encrypted in the Preparation exit before sending it across insecure network connections. If the `PASSWORD` is needed in the application again, it must be decrypted after receipt in the Evaluation exit. The authentication exit must ensure that the `PASSWORD` is properly decrypted if necessary before sending it to an external security system.

The EntireX Broker provides minimal encryption of the `PASSWORD` field, that is, the field is not transmitted in plain text. If your environment requires absolute security, however, you will need to provide both Broker stub and EntireX Broker kernel exits to perform encryption and decryption. See the description of the `PASSWORD` field in the Broker ACI control block.

SECURITY-TOKEN

The `SECURITY_TOKEN` can be created by the application and sent to EntireX Broker. That allows for a kind of credential algorithm. The security token is passed to all kernel exits and can therefore contain security information which is also important for the authorization and encryption exits. The `SECURITY_TOKEN` can be altered in the authentication exit to provide a context token for that application and that session. It is transmitted back to the application and can then be used in all subsequent calls. For each subsequent call, the EntireX Broker checks whether the `SECURITY_TOKEN` is identical to the one returned from the last call to the authentication exit, which could be the `ETB_SEC_LOGON`, the `ETB_SEC_NEWPUI` or the `ETB_SEC_NEWST` function. After an `ETB_SEC_LOGOFF` call, a subsequent call is always a `ETB_SEC_LOGON` call. See the description of the `SECURITY-TOKEN` field in the Broker ACI control block.

CLIENT-UID

`CLIENT-UID` is returned to a server application after a `RECEIVE` and contains the user ID of the sending client. This allows for further security checks (logging, separate checks, etc.). See the description of the `CLIENT-UID` field in the Broker ACI control block.

ERROR-CODE

All security-related `ERROR_CODES` start with the `ERROR_CLASS 0008`. The following four digits in the `ERROR_CODE` can be assigned by any exit if a security violation occurs. These digits only reach the application if the current operation is aborted by the security exit with a security violation, i.e. an appropriate return code. See `ERROR-CODE` under *Broker ACI Fields*.

ERROR-TEXT

The security exits can also pass an error message back to the application. This error text must not be longer than 40 bytes.

KERNELSECURITY

See `KERNELSECURITY` under *Broker ACI Fields*.

Programming Broker Stub Exits

The exits in the stub have the following interface:

- [Preparation Exit](#)
- [Evaluation Exit](#)
- [Programming the Kernel Exit Routine](#)

Preparation Exit

Synopsis

```
int etbupre (ETBCB *pEtbCb,
            void *pSendBuf,
            void *pReserved,
            char *pErrText)
```

Parameter	Format	Direction	Description
Address of ETBCB	Pointer to ETBCB control block.	I/O	ETBCB's user_id and password are used to generate the credential which will be saved in field security_token for function LOGON.
Address of send buffer	void pointer	I/O	Send buffer ⁽³⁾ supplied by caller, only available for function SEND, length of send buffer is member of ETBCB.
Reserved	void pointer	I	Must be NULL.
Address of error text	char pointer	O	The error text is an array of 40 characters containing the error text that will be returned by the stub routine.

Return value

0 (okay) or non-zero (error)

The real error code must be written to the Broker control block as an 8-byte character array (without trailing 0 byte!). The *Message Class 0008 - EntireX ACI - Security Error* is reserved for all errors in function etbupre. The error number is user-defined. Additionally, the error text can be returned to the user in the error text array.

Required Actions in the Exit

If no data encryption is desired, no action is required.

Possible Actions in the Exit

- Generate a credential if function is `LOGON` and move it to the field `security_token`.
- Encrypt the send buffer if function is `SEND`. The encryption process must not change the length of the buffer. See Notes below.

The exit gets control for each function of ACI version 2 and above. The exit must exist.



Notes:

1. We recommend *not* implementing your own encryption/decryption mechanism. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS and Certificates with EntireX*.
2. The mainframe stubs (e.g. z/OS, BS2000) employ high performance direct cross memory to send and receive data from buffers in the application's working storage. This is utilized for sending/receiving more than 32 KB of data. Therefore, when encryption is active, the application programmer must not rely on the contents of the `SEND` buffer after issuing the `SEND` command, since the contents of the `SEND` buffer will be encrypted when sending more than 32 KB of data. We recommend you code all applications so that you do not rely on the contents of the `SEND` buffer after calling Broker. This will be required in the future for all `SEND` commands regardless of whether the data exceeds 32 KB. Therefore, the application's `SEND` buffer must not be in read-only memory, where encryption is activated.
3. A single send/receive buffer is used to perform encryption in place. This means that encrypted data is provided in the send buffer.

Evaluation Exit

Synopsis

```
int etbueva (ETBCB *pEtbCb,  
            void *pRecBufEncr,  
            void *pReserved,  
            char *pErrText)
```


Parameters

Parameter	Format	Direction	Description
Address of ETBCB	Pointer to ETBCB control block.	I/O	ETBCB's security token is used for data decryption.
Address of receive buffer	void pointer	I/O	Receive buffer ⁽²⁾ provided by EntireX Broker. Only available for functions <code>RECEIVE</code> and <code>SEND WAIT=x</code> (implicit receive). Length of receive buffer is member of ETBCB.
Reserved	void pointer	I	Must be NULL.
Address of error text	char pointer	O	The error text is an array of 40 characters containing the error text which will be returned by the stub routine.

Return Value

0 (okay) or non-zero (error)

The real error code must be written to the Broker control block as an 8-byte character string (without trailing 0 bytes!). The *Message Class 0008 - EntireX ACI - Security Error* is reserved for all errors in function `etbueva`. The error number is user-defined.

In addition, the error text can be returned to the user.

Required Actions in the Exit

If no data decryption is wanted, no action is required.

Possible Actions in the Exit

- Decrypt ⁽¹⁾ the receive buffer if functions are `RECEIVE` or `SEND` with `WAIT`. The decryption process must not change the length of the buffer.

The exit gets control for each function of ACI Version 2 and above. The exit must exist.



Notes:

1. We recommend *not* implementing your own encryption/decryption mechanism. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS and Certificates with EntireX*.
2. A single send/receive buffer is used to perform decryption in place. This means that decrypted data is provided in the receive buffer. After the completion of a send/nawait command, the application should ignore the contents of the receive buffer.

Programming the Kernel Exit Routine

All authentication, authorization, encryption ⁽¹⁾ and decryption exits are combined within one exit module named `USRSEC`. The various security checks are indicated by a type parameter in the `CALL` interface. `USRSEC` is provided with EntireX Broker as the default security exit. It is invoked if `SECURITY=YES` is set in the attribute file. Prior to EntireX, the `USRSEC` exit was available only with the SAF Gateway security package.

The general syntax of this user exit is defined as follows:

Synopsis

```
long usrsec (ETB_SECPAR *pParSec,
             void *pVarious,
             char *pErrText,
             char *pWorkArea,
             long lWorkArea)
```

Parameters

Parameter	Format	Direction	Description
Address of security parameter block	Pointer to structure <code>ETB_SECPAR</code>	I	Contains the security type flag.
Address of type-dependent security parameter block	void pointer	I	See control block structures <code>ETB_SECPAR_<type></code> .
Address of error text	char pointer	O	The error text is an array of 40 characters containing the error text which will be returned to the user.
Address of work area	char pointer	O	Volatile work area.
Length of work area	long integer value	I	Size of the work area in number of bytes.

Return Value

0 (okay) or user-defined error number

Message Class 0008 - EntireX ACI - Security Error and the error number will be returned to the user. In addition, the error text can be returned to the user.



Notes:

1. We recommend *not* implementing your own encryption/decryption mechanism. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS and Certificates with EntireX*.

Required/Recommended Actions in the Exit (depending on Security Type)

Security Type	Required Action	Recommended Action	Notes
ETB_SEC_ENCRYPT	Copy decrypted to encrypted buffer and set the length of encrypted buffer. This is necessary because exit is called whether the receive data has to be encrypted or not.	Encrypt receive data if needed.	1,2
ETB_SEC_DECRYPT	Copy encrypted to decrypted buffer and set the length of decrypted buffer. This is necessary because exit is called irrespective of whether send data is encrypted or not.	Decrypt receive data if needed.	1,2
ETB_SEC_LOGON		Decrypt the password and check combination of user ID and password against the security system. Generate a context token according to the credentials of the user and EntireX Broker. Create a security handle for a user session (e.g. ACEE on z/OS).	
ETB_SEC_LOGOFF	None	Delete the security handle of the user session.	
ETB_SEC_NEWPUID	None	An application has changed the physical user ID between two calls. If necessary, a new security token can be created.	
ETB_SEC_NEWST	None	For some reason, the security token of an application has changed and no longer matches the original. The security token should be recalculated and approved or the application should be rejected.	
ETB_SEC_REGISTER	None	Check whether user_id is authorized to offer the requested SERVICE (check security_token data if necessary).	
ETB_SEC_SEND	None	Check whether user_id is authorized to offer the requested SERVICE (check security_token data if necessary).	
ETB_SEC_SECURE	None	Check the security token.	
ETB_SEC_INIT	None	Create global context if required.	
ETB_SEC_FINI	None	Delete global context if created.	
ETB_SEC_INFO	None	Check CIS information request.	
ETB_SEC_CMD	None	Check CIS command request.	



Notes:

1. The size of the buffer cannot be changed in this exit.
2. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS and Certificates with EntireX*.

V

■ 9 EntireX Broker Reporting	139
■ 10 Command Logging in EntireX	149

9 EntireX Broker Reporting

■ Configuration Report	140
■ Load Module Report	141
■ Storage Report	142
■ Persistent Store Report	145
■ License Report	147

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 UNIX 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  VRSPB Build number      Alias  Level  CurNo
      Steplib level 0: SAG.EXB731.LOAD
      1  ADAACK
      2  ADABSP
      3  ADACDC
      4  ADACLU
      5  ADACLX
      6  ADACMO
      7  ADACMP
      8  ADACMR
      9  ADACMU
     10  ADACNS
     11  ADACNV
      ...
     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
     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.

UNIX 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.

z/VSE

Logical unit `SYS015` and logical file name *ETBSREP* are used. Format `RECORD-FORMAT=FB`, `RECORD-LENGTH=121` is used.

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 ↵
KERNEL POOL		0x25E48010	407184 bytes	407184 bytes	2009-06-26 ↵
12:28:58.768	Allocated				
HEAP POOL		0x25EB4010	1050692 bytes	1457876 bytes	2009-06-26 ↵
12:28:58.769	Allocated				
COMMUNICATION POOL		0x25FB5010	16781380 bytes	18239256 bytes	2009-06-26 ↵
12:28:58.769	Allocated				
ACCOUNTING POOL		0x26FB7010	762052 bytes	19001308 bytes	2009-06-26 ↵
12:28:58.769	Allocated				
BROKER POOL		0x27072010	61540 bytes	19062848 bytes	2009-06-26 ↵
12:28:58.775	Allocated				
CONVERSATION POOL		0x27082010	368964 bytes	19431812 bytes	2009-06-26 ↵
12:28:58.775	Allocated				
CONNECTION POOL		0x270DD010	233668 bytes	19665480 bytes	2009-06-26 ↵
12:28:58.779	Allocated				
LONG MESSAGES POOL		0x27117010	4395204 bytes	24060684 bytes	2009-06-26 ↵
12:28:58.782	Allocated				
SHORT MESSAGES POOL		0x27549010	3703876 bytes	27764560 bytes	2009-06-26 ↵
12:28:58.806	Allocated				
PARTICIPANT POOL		0x278D2010	134244 bytes	27898804 bytes	2009-06-26 ↵
12:28:58.827	Allocated				
PARTICIPANT EXTENSION POOL		0x278F3010	36996 bytes	27935800 bytes	2009-06-26 ↵
12:28:58.829	Allocated				
PROXY QUEUE POOL		0x278FD010	26724 bytes	27962524 bytes	2009-06-26 ↵
12:28:58.829	Allocated				
SERVICE ATTRIBUTES POOL		0x27904010	131668 bytes	28094192 bytes	2009-06-26 ↵
12:28:58.829	Allocated				
SERVICE POOL		0x27925010	54372 bytes	28148564 bytes	2009-06-26 ↵
12:28:58.830	Allocated				
SERVICE EXTENSION POOL		0x27933010	32900 bytes	28181464 bytes	2009-06-26 ↵
12:28:58.831	Allocated				
TIMEOUT QUEUE POOL		0x2793C010	87268 bytes	28268732 bytes	2009-06-26 ↵
12:28:58.831	Allocated				
TRANSLATION POOL		0x27952010	179300 bytes	28448032 bytes	2009-06-26 ↵
12:28:58.832	Allocated				
UNIT OF WORK POOL		0x2797E010	176324 bytes	28624356 bytes	2009-06-26 ↵
12:28:58.834	Allocated				
WORK QUEUE POOL		0x279AA010	391268 bytes	29015624 bytes	2009-06-26 ↵
12:28:58.835	Allocated				
BLACKLIST POOL		0x27A0A010	42084 bytes	29057708 bytes	2009-06-26 ↵
12:28:58.838	Allocated				
COMMUNICATION POOL		0x25FB5010	16781380 bytes	12837332 bytes	2009-06-26 ↵
12:30:58.514	Deallocated				
ACCOUNTING POOL		0x26FB7010	762052 bytes	12075280 bytes	2009-06-26 ↵

12:30:58.515	Deallocated					
BROKER POOL		0x27072010	61540 bytes	12013740 bytes	2009-06-26	↔
12:30:58.516	Deallocated					
CONVERSATION POOL		0x27082010	368964 bytes	11644776 bytes	2009-06-26	↔
12:30:58.518	Deallocated					
CONNECTION POOL		0x270DD010	233668 bytes	11411108 bytes	2009-06-26	↔
12:30:58.519	Deallocated					
LONG MESSAGES POOL		0x27117010	4395204 bytes	7015904 bytes	2009-06-26	↔
12:30:58.520	Deallocated					
SHORT MESSAGES POOL		0x27549010	3703876 bytes	3312028 bytes	2009-06-26	↔
12:30:58.526	Deallocated					
PROXY QUEUE POOL		0x278FD010	26724 bytes	3285304 bytes	2009-06-26	↔
12:30:58.530	Deallocated					
TIMEOUT QUEUE POOL		0x2793C010	87268 bytes	2690464 bytes	2009-06-26	↔
12:30:58.532	Deallocated					
UNIT OF WORK POOL		0x2797E010	176324 bytes	2514140 bytes	2009-06-26	↔
12:30:58.533	Deallocated					
WORK QUEUE POOL		0x279AA010	391268 bytes	2122872 bytes	2009-06-26	↔
12:30:58.533	Deallocated					
BLACKLIST POOL		0x27A0A010	42084 bytes	2080788 bytes	2009-06-26	↔
12:30:58.534	Deallocated					
PARTICIPANT POOL		0x278D2010	134244 bytes	1893112 bytes	2009-06-26	↔
12:49:25.817	Deallocated					
PARTICIPANT EXTENSION POOL		0x278F3010	36996 bytes	1856116 bytes	2009-06-26	↔
12:49:25.818	Deallocated					
SERVICE ATTRIBUTES POOL		0x27904010	131668 bytes	1724448 bytes	2009-06-26	↔
12:49:25.818	Deallocated					
SERVICE POOL		0x27925010	54372 bytes	1670076 bytes	2009-06-26	↔
12:49:25.818	Deallocated					
SERVICE EXTENSION POOL		0x27933010	32900 bytes	1637176 bytes	2009-06-26	↔
12:49:25.819	Deallocated					
TRANSLATION POOL		0x27952010	179300 bytes	1457876 bytes	2009-06-26	↔
12:49:25.819	Deallocated					
HEAP POOL		0x25EB4010	1050692 bytes	407184 bytes	2009-06-26	↔
12:49:25.820	Deallocated					
KERNEL POOL		0x25E48010	407184 bytes	0 bytes	2009-06-26	↔
12:49:25.820	Deallocated					

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.

UNIX

Broker creates a file with the name *PSTORE.REPORT* in the current working directory. The file name *PSTORE.REPORT.LOAD* will be used if Broker is started with `RUN-MODE = PSTORE-LOAD`.

The file name *PSTORE.LOAD.UNLOAD* will be used if Broker is started with `RUN-MODE = PSTORE-UNLOAD`.

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 UNIX.

z/OS

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

z/VSE

Logical unit `SYS003` and logical file name `ETBPREP` are used. Format `RECORD-FORMAT = FB`, `RECORD-LENGTH = 121` is used.

Sample Report

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

EntireX 10.1	PSTORE Report	2016-10-18 10:46:18	Page	1		
Identifier	Elements	Total length	Record Type	Date	Time	↔
Action	Remaining Postpone Attempts					
0000000000000000	1	760	Master	2016-10-18	10:46:18.529	↔
Created						
0010000000000001	1	5022	Conversation	2016-10-18	10:46:36.216	↔
Created						
0010000000000002	1	5022	Conversation	2016-10-18	10:46:36.216	↔
Created						
0010000000000003	1	5022	Conversation	2016-10-18	10:46:36.217	↔
Created						
0010000000000001			Conversation	2016-10-18	10:46:48.036	↔
Postponed	6					
0010000000000001			Conversation	2016-10-18	10:47:18.470	↔
Accepted	6					
0010000000000002			Conversation	2016-10-18	10:47:57.856	↔
Postponed	6					
0010000000000002			Conversation	2016-10-18	10:48:27.476	↔
Accepted	6					
0010000000000003			Conversation	2016-10-18	10:48:27.477	↔
Postponed	6					
0010000000000003			Conversation	2016-10-18	10:48:57.480	↔
Accepted	6					
0010000000000003			Conversation	2016-10-18	10:48:57.481	↔
Postponed	5					
0010000000000003			Conversation	2016-10-18	10:49:27.484	↔
Accepted	5					
0010000000000001			Conversation	2016-10-18	10:52:25.505	↔
Deleted						
0010000000000002			Conversation	2016-10-18	10:52:25.506	↔
Deleted						
0010000000000003			Conversation	2016-10-18	10:53:26.508	↔
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
 - **Loaded**: record is loaded (Broker instance with RUN-MODE = PSTORE-LOAD)
 - **Unloaded**: record is unloaded (Broker instance with RUN-MODE = PSTORE-UNLOAD)
- **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.

10

Command Logging in EntireX

■ Introduction to Command Logging	150
■ Command Log Filtering using Command-line Interface ETBCMD	152
■ ACI-driven Command Logging	154
■ Dual Command Log Files	154

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.

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
<code>CMDLOG</code>	Set this to "N" if command logging is not needed.
<code>CMDLOG-FILE-SIZE</code>	A numeric value indicating the maximum size of command log file in KB.
<code>NUM-CMDLOG-FILTER</code>	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 UNIX 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.

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.

UNIX and Windows

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

z/OS

Command	Description
<code>//ETBCMD EXEC PGM=ETBCMD, // PARM=('/-blocalhost:1970:TCP ↵ -cSET-CMDLOG-FILTER -xuser ', // '-dBROKER ↵ -nAClass/ASERVER/ASERVICE')</code>	This command sets filters on ACLASS/ASERVER/ASERVICE. All ACI calls issued by <i>all</i> users to this service will be logged.
<code>//ETBCMD EXEC PGM=ETBCMD, // PARM=('/-blocalhost:1970:TCP ↵ -cSET-CMDLOG-FILTER -xuser ', // '-dBROKER -nAClass/ASERVER/ASERVICE ↵ -Usaguser1')</code>	This command sets filters on ACLASS/ASERVER/ASERVICE and user ID saguser1. All ACI calls to this service <i>as well as</i> 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 UNIX:

```
etbcd -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 UNIX:

```
etbcd -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 UNIX:

```
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.

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 the variables `ETB_CMDLOG1` and `ETB_CMDLOG2` as the default command log file names.

Under UNIX, the startup script uses file names `CMDLOGR1` and `CMDLOGR2`.

Under Windows, the keys `ETB_CMDLOG1` and `ETB_CMDLOG2` are entered in the Registry with values `CMDLOGR1` and `CMDLOGR2`.

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 UNIX:

```
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.

