webMethods EntireX

Administration under UNIX

Version 10.5

October 2019

WEBMETHODS

This document applies to webMethods EntireX Version 10.5 and all subsequent releases.

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

Document ID: EXX-ADMIN-105-20220422UNIX

Table of Contents

EntireX Administration under UNIX

Broker Configuration	Broker-related configuration topics.
Broker Add-ons	Broker add-ons: Broker stubs, command-line utilities.
Broker Agents	Broker Agents.
RPC Servers and Listeners	RPC servers and listeners under UNIX.
Logging and Tracing EntireX	Logging, tracing and accounting.

1 About this Documentation

Document Conventions

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

Online Information and Support

Product Documentation

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

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

Product Training

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

Tech Community

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

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

Product Support

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

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

Data Protection

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

2 Setting up Broker Instances

This chapter contains information on setting up the Broker under UNIX. It assumes that you have successfully installed EntireX using the Software AG Installer.

Startup Daemon for Broker Administration

When installation is complete, the daemon is running and ready to be used by `etbsrv` script located in directory *<Installation_Dir>/EntireX/bin*. This script can be used, for example, to start or stop the broker.

To start the daemon

- Enter the following command:

```
- <Installation_Dir>/EntireX/bin/sagexx105 start
```

To stop the daemon

- Enter the following command:

```
- <Installation_Dir>/EntireX/bin/sagexx105 stop
```

It is also registered to startup at boot time, therefore the installation generates additional scripts in a location that depends on the operating system:

Operating System	Location	Note
Solaris, Linux	/etc/init.d	Recent Linux versions use systemd instead of init scripts.
AIX	/etc	

Setting up the TCP/IP Communication

The recommended way to set up TCP/IP is to define attribute `PORT=nnnn` and optionally `HOST=x.x.x.x|hostname` in the TCP-specific section of the broker attribute file.

If no port number is specified, the EntireX Broker kernel uses `getservbyname` to determine the TCP/IP port on which it will listen for incoming connections. The specified name is the value of `BROKER-ID` in the attribute file.

An entry for this value must be made in the local machine's */etc/services* file, for example:

```
etbnnn yyyyy/tcp # local host
```

where `etb`*`nnn`* is the `BROKER-ID` and

`yyyyy` is the intended port number.

This is the same place that local broker stubs will obtain the port information. If `getservbyname` fails, the default port number 1971 is used. This is the same default port number that the stubs use.

Starting and Stopping the Default Broker

If check box **Turn on Autostart for default EntireX Broker** is checked during installation, the default broker ETB001 is started.

To start the default broker

- Enter command:

```
<Installation_Dir>/EntireX/bin/defaultbroker start
```

To stop the default broker

- Enter command:

```
<Installation_Dir>/EntireX/bin/defaultbroker stop
```

Running Broker with SSL/TLS Transport

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* in the Platform-independent Administration documentation.

Before starting the Broker, it must be configured to correctly use SSL as a transport mechanism:

To set up SSL

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

2 Modify broker-specific attributes.

Append "`-SSL`" to the `TRANSPORT` attribute. For example:

```
DEFAULTS = BROKER
TRANSPORT = TCP-SSL
```

See also `TRANSPORT`.

3 Set the SSL attributes, for example:

```
DEFAULTS = SSL
 KEY-STORE = /<Install_Dir>/EntireX/etc/ExxAppCert.pem
 KEY-PASSWD-ENCRYPTED = MyAppKey
 KEY-FILE  = /<Install_Dir>/EntireX/etc/ExxAppKey.pem
 VERIFY-CLIENT = N
 PORT=1958
```

where *1958* is the default but can be changed to any port number.

See also *SSL/TLS-specific Attributes* and *SSL/TLS and Certificates with EntireX*.

4 Make sure the SSL clients connecting to the broker are prepared for SSL connections as well. See *Using SSL/TLS with EntireX Components*.

Starting and Stopping an Additional Broker

A default broker is always created during installation. This broker is started automatically by default.

1. Create a subdirectory called ETB*nnn* under *$EXXDIR/etb* if it does not yet exist, place the attribute file under ETB*nnn* and name it *etbfile*.

   Example:

```
cd $EXXDIR/etb
mkdir ETB002
cp /tmp/your attribute file ETB002/etbfile
```

2. The broker can be started by executing shell script *etbstart* in the /<*Install_Dir*>/*EntireX*/*bin* directory, using the syntax:

   ```
   etbstart ETBnnn
   ```

   where ETB*nnn* is the assigned Broker ID (for example ETB001).

3. The broker can be stopped by executing the *etbcmd* utility in the /<*Install_Dir*>/*EntireX*/*bin* directory using the syntax:

   ```
   etbcmd -bbroker-id -dBROKER -cSHUTDOWN
   ```

   Optional: The broker can also be shut down in any of the following ways:

   - `etbcmd -blocalhost:port -dBROKER -cSHUTDOWN`
   - `etbcmd -bipaddress:port -dBROKER -cSHUTDOWN`
   - `etbcmd -bmachinename:port -dBROKER -cSHUTDOWN`

   The port number is needed only when the broker is not running on the standard port.

   See also ***Broker Shutdown Statistics*** and ***Setting up TCP/IP Transport***.

Note: The information given here is independent of hardware type and platform.

Uniqueness Test for Broker ID

To guarantee that a broker ID is unique on one machine, a named semaphore is created at initialization. If this semaphore already exists for this broker ID, initialization is terminated with message ETBE0168, “This instance of broker already running”. If as a result of an abnormal broker termination this semaphore cannot be deleted completely, you can force a restart of the Broker with Broker attribute `FORCE=YES`.

Tracing EntireX Broker

This section covers the following topics:

Broker TRACE-LEVEL Attribute

The Broker `TRACE-LEVEL` attribute determines the level of tracing to be performed while Broker is running. The Broker has a master `TRACE-LEVEL` specified in the Broker section of the attribute file as well as several individual `TRACE-LEVEL` settings that are specified in the following sections of the attribute file.

Individual Settings	Specified in Attribute File Section	Note
Master trace level	`DEFAULTS=BROKER`	1,2
Persistent store trace level	`DEFAULTS=ADABAS | CTREE | DIV`	1
Conversion trace level	`DEFAULTS=SERVICE`; Trace option of the service-specific broker attribute `CONVERSION`.	
Security trace level	`DEFAULTS=SECURITY`	1
Transport trace level	`DEFAULTS=TCP | SSL`	1
Application Monitoring trace level	`DEFAULTS=APPLICATION-MONITORING`	

Notes:

1. For temporary changes to the master or individual `TRACE-LEVEL` without restarting the Broker, use the Broker command-line utility `etbcmd`.
2. For temporary changes to the master `TRACE-LEVEL` without restarting the broker, use Command Central. See *Changing the Trace Level Temporarily*.

Attribute File Trace Setting

Trace Level	Description
0	No tracing. Default value.
1	Traces incoming requests, outgoing replies, and resource usage.
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.

Note: Trace levels 2 and above should be used only when requested by Software AG support.

Deferred Tracing

It is not always convenient to run with `TRACE-LEVEL` defined, especially when higher trace levels are involved. Deferred tracing is triggered when a specific condition occurs, such as an ACI response code or a broker subtask abend. Such conditions cause the contents of the trace buffer to be written, showing trace information leading up the specified event. If the specified event does not occur, the Broker trace will contain only startup and shutdown information (equivalent to `TRACE-LEVEL=0`). Operating the trace in this mode requires the following additional attributes in the broker section of the attribute file. Values for `TRBUFNUM` and `TRAP-ERROR` are only examples.

Attribute	Value	Description
`TRBUFNUM`	`3`	Specifies the deferred trace buffer size = 3 * 64 K.
`TRMODE`	`WRAP`	Indicates trace is not written until an event occurs.
`TRAP-ERROR`	`322`	Assigns the event ACI response code 00780322 "PSI: UPDATE failed".

Dynamically Switching On or Off the EntireX Broker Trace

The following methods are available to switch on or off the EntireX Broker trace dynamically. You do not need to restart the broker for the changes to take effect.

- `etbcmd`
  Run command utility `etbcmd` with option `-c TRACE-ON or - c TRACE-OFF`. See `etbcmd`.
- **Command Central**
  Use Command Central. See *Updating the Trace Level* under *Administering the EntireX Broker* using the Command Central GUI | Command Line.

Trace File Handling

Attributes MAX-TRACE-FILES and TRACE-FILE-SIZE are used to avoid a constantly growing ETB.LOG file. The trace is written to file ETB.LOG until TRACE-FILE-SIZE has been reached and a new file is opened. The number of files defined in MAX-TRACE-FILES is kept in addition to the current ETB.LOG file.

Example: If you define MAX-TRACE-FILES=9 and TRACE-FILE-SIZE=100M, the current ETB.LOG will be closed after 100 MB have been written. A maximum of nine backup files plus the current ETB.LOG file are kept.

Protecting a Broker against Denial-of-Service Attacks

An optional feature of EntireX Broker is available to protect a broker running with SECURITY=YES against denial-of-service attacks. An application that is running with invalid user credentials will get a security response code. However, if the process is doing this in a processing loop, the whole system could be affected. If PARTICIPANT-BLACKLIST is set to YES, EntireX Broker maintains a blacklist to handle such “attacks”. If an application causes ten consecutive security class error codes within 30 seconds, the blacklist handler puts the participant on the blacklist. All subsequent requests from this participant are blocked until the BLACKLIST-PENALTY-TIME has elapsed.

Server Shutdown Use Case

Here is a scenario illustrating another use of this feature that is not security-related.

An RPC server is to be shut down immediately, using Broker Command and Information Services (CIS), and has no active request in the broker. The shutdown results in the LOGOFF of the server. The next request that the server receives will probably result in message 00020002 "User does not exist", which will cause the server to reinitialize itself. It was not possible to inform the server that shutdown was meant to be performed.

With the *blacklist*, this is now possible. As long as the blacklist is not switched off, when a server is shut down immediately using CIS and when there is no active request in the broker, a marker is set in the blacklist. When the next request is received, this marker results in message 00100050 "Shutdown IMMED required", which means that the server is always informed of the shutdown.

3 Configuring the Administration Service

The Administration Service allows you to start, stop, and retrieve the status of a local broker.

It is provided in a fully functional state and is started by the installation. It needs access to local port 57909 with restriction to local users.

The port of the Administration Service can be changed in the configuration file *etc/brokeradminwrapper.conf*. If you change the port you need to restart the administration service. This is the line to change:

```
wrapper.java.additional.101=-Dcom.sun.management.jmxremote.port=57909
```

In most cases you will not need to change the configuration file. The log files provide more information about the service and can help you analyze the cause of any error that occurs. The log file is called *wrapper.log* and is located in *config/etb*.

The Administration Service requires SSL certificates to create brokers with SSL ports. These certificates are for test purposes only and constitute a security risk. If you want to use SSL, replace the certificates in the *etc* directory with your own SSL certificates.

When a broker is created, the Administration Service copies the required certificates from the EntireX *etc* directory to the working directory of the newly created broker.

If the certificates are to be replaced after the installation, you also need to replace the certificates in the working directories ETB001 (Default Broker) and in the EntireX directory *etc*.

4 Broker Attributes

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
UNIX	File *etbfile* in directory <*InstDir*>/EntireX/config/etb/<*BrokerName*> (default) *

* 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 `${variable name}` is replaced with the value of the corresponding environment variable.
- On z/OS, variable values are read from a file defined by the DD name `ETBVARS`. The syntax of this file is the same as the attribute file.
- If a variable has no value: if the variable name is enclosed in braces, error 00210594 is given, otherwise `$variable name` will be used as the variable value.
- If you encounter problems with braces (and this is quite possible in a z/OS environment), we suggest you omit the braces.

Broker-specific Attributes

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

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

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
ACCOUNTING-VERSION	1 | 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.						
ACI-CONVERSION	YES | NO	O	z	u	w	v	b
	Determines the handling of ACI request and response strings of USTATUS. YES Convert ACI request and response strings with ICU. See *ICU Conversion* in the Internationalization documentation. NO Translate ACI request and response with internal translation table without support of national characters. See *Translation User Exit* in the Internationalization documentation. **Note:** This attribute was undocumented in earlier EntireX versions and had default value NO. This meant that a translation user exit was used instead; this is no longer recommended.						
APPLICATION-MONITORING or APPMON	YES | NO	O	z	u	w	v	b
	Enable application monitoring in EntireX Broker. YES Enable application monitoring. NO Disable application monitoring. See *Application Monitoring*.						
AUTOLOGON	YES | 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.						
AUTOSTART	NO | YES	O		u	w		
	This attribute defines the autostart behavior of a broker. NO Broker is *not* started automatically with the next system start. YES Broker is restarted automatically with the next system start.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Note: Prior to EntireX version 10.5 this was handled by the Broker Administration Service.						
BLACKLIST-PENALTY-TIME	5m | *n* | *n*S | *n*M | *n*H	R	z	u	w	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 *n*S. *n*S Non-activity time in seconds (max. 2147483647). *n*M Non-activity time in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). See *Protecting a Broker against Denial-of-Service Attacks* in the platform-specific 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	15M | *n* | *n*S | *n*M | *n*H	R	z	u	w	v	b
	Define the non-activity time for clients. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647). *n*M Non-activity time in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). A client that does not issue a broker request within the specified time limit is treated as inactive and all resources for the client are freed.						
CMDLOG	NO | YES	O	z	u	w	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	1024 | *n*	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*.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
CONTROL-INTERVAL	60s | *n* | *n*S | *n*M | *n*H |	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. 2. The standby broker will check the status of the standard broker after the elapsed CONTROL-INTERVAL time. *n* Same as *n*S. *n*S Interval in seconds (max. 2147483647). *n*M Interval in minutes (max. 35791394). *n*H Interval in hours (max. 596523). The minimum value is 16 seconds. We strongly recommend the default value (60 seconds), except for very slow machines.						
CONV-DEFAULT	UNLIM | *n*	O	z	u	w	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. *n* Number of conversations. This value can be overridden by specifying a CONV-LIMIT for the service. A value of 0 (zero) is invalid.						
DEFERRED	NO | YES	O	z	u	w	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	YES | 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 *Dynamic Memory Management* under *Broker Resource Allocation*. 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
	If you run your broker with attribute DYNAMIC-MEMORY-MANAGEMENT=YES, the following attributes are not needed: ■ CONV-DEFAULT ■ HEAP-SIZE ■ LONG-BUFFER-DEFAULT ■ SERVER-DEFAULT ■ SHORT-BUFFER-DEFAULT ■ NUM-CLIENT ■ NUM-CMDLOG-FILTER ■ NUM-COMBUF ■ NUM-CONV[ERSATION] ■ NUM-LONG[-BUFFER] ■ NUM-SERVER ■ NUM-SERVICE-EXTENSION ■ NUM-SERVICE ■ NUM-SHORT[-BUFFER] ■ NUM-UOW|MAX-UOWS|MUOW ■ NUM-WQE **Caution:** However, if one of these attributes is defined, it determines the allocation size of that particular broker resource.						
DYNAMIC-WORKER-MANAGEMENT	NO | YES	O	z	u	w		b
	NO All worker tasks are started at broker startup. The number of worker tasks is defined by NUM-WORKER. After this initial step, no further worker tasks can be started. This is default and simulates the behavior of EntireX version 8.0 and earlier. YES As above, the initial portion of worker tasks started at broker startup is determined by NUM-WORKER. However, if there is a need to handle an increased workload, additional worker tasks can be started at runtime without restarting broker. Conversely, if a worker task remains unused, it is stopped. The upper and lower limit of running worker tasks can be defined by the attributes WORKER-MIN and WORKER-MAX. If you run broker with DYNAMIC-WORKER-MANAGEMENT=YES, the following attributes are useful to optimize the overall processing: ■ WORKER-MAX ■ WORKER-MIN ■ WORKER-NONACT ■ WORKER-QUEUE-DEPTH ■ WORKER-START-DELAY The attribute NUM-WORKER defines the initial number of worker tasks started during initialization. See *Dynamic Worker Management* under *Broker Resource Allocation*.						

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	YES The broker tries to locate ICU custom converters with the mechanism defined by the platform, see *Building and Installing ICU Custom Converters* in the platform-specific Administration documentation. NO Use of ICU custom converters is not possible.						
IPV6	YES | NO	O	z	u	w		b
	YES Establish SSL and TCP/IP transport in IPv6 and IPv4 networks according to the TCP/IP stack configuration. NO Establish SSL and TCP/IP transport in IPv4 network only. This attribute applies to EntireX version 9.0 and above.						
LONG-BUFFER-DEFAULT	UNLIM | *n*	O	z	u	w	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. *n* Number of buffers. This value can be overridden by specifying a LONG-BUFFER-LIMIT for the service. A value of 0 (zero) is invalid.						
MAX-MEMORY	0 | *n* | *n*K | *n*M | *n*G | UNLIM	O	z	u	w	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.						
MAX-MESSAGE-LENGTH	2147483647 | *n*	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	16 | *n*	O	z	u	w	v	b
	Maximum number of messages in a UOW.						
MAX-MSG	See MAX-MESSAGE-LENGTH.						
MAX-TRACE-FILES	4 | *n*	O		u	w		
	Defines the number of backup copies of the trace file ETB.LOG. Minimum number is 1; maximum is 999. A new trace file is allocated when the value for TRACE-FILE-SIZE is exceeded. These two attributes prevent a constantly growing						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	ETB.LOG file. See *Trace File Handling* in the UNIX and Windows Administration documentation.						
MAX-UOW-MESSAGE-LENGTH	See MAX-MESSAGE-LENGTH.						
MAX-UOWS	0 | *n*	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	NONE | 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	YES | NO	O	z	u	w	v	b
	YES New UOW messages are allowed. NO New UOW messages are not allowed. This applies to UOW when using Persistence and should not be used for non-persistent UOWs. A usage example could be the following: The broker persistent store reaches capacity and the broker shuts down. You can set NEW-UOW-MESSAGES to NO to prevent new UOW messages from being added after a broker restart. This action allows only consumption (not production) of UOWs to occur after broker restart. After the persistent store capacity has been sufficiently reduced, the EntireX Broker administrator can issue a CIS command, see ALLOW-NEWUOWMSGS. This action allows new UOW messages to be sent to the broker. Reset attribute NEW-UOW-MESSAGES to YES, which permits new UOW messages to be produced in subsequent broker sessions.						
NUM-BLACKLIST-ENTRIES	256 | *n*	O	z	u	w	v	b
	Number of entries in the participant blacklist. Default value is 256 entries. Together with BLACKLIST-PENALTY-TIME and PARTICIPANT-BLACKLIST, this attribute is used to protect a broker running with SECURITY=YES against denial-of-service attacks. See *Protecting a Broker against Denial-of-Service Attacks* in the platform-specific broker Administration documentation.						
NUM-CLIENT	*n*	R	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Number of clients that can access the broker concurrently. A value of 0 (zero) is invalid.						
NUM-CMDLOG-FILTER	1 | *n*	O	z	u	w	v	b
	Maximum number of filters that can be specified simultaneously. **Tip:** We recommend you limit this value to the number of services that are being monitored. Minimum value is 1. A value of zero is invalid when the attribute CMDLOG is set to YES. See *Command Logging in EntireX* for more information.						
NUM-COMBUF	1024 | 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	*n* | 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.) *n* Number of conversations. 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. **Note:** 1. A value of 0 (zero) is invalid. If a wildcard service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 2. See *Wildcard Service Definitions*.						
NUM-LONG-BUFFER or NUM-LONG	4096 | *n* | AUTO	R	z	u	w	v	b
	Defines the number of long message containers. Long message containers have a fixed length of 4096 bytes and are used to store requests that are larger than 2048 bytes. Storing a request of 8192 bytes, for example, would require two long message containers. *n* Number of buffers. AUTO Uses the LONG-BUFFER-DEFAULT and the service-specific LONG-BUFFER-LIMIT values to calculate the number of long message buffers. The values used in the calculation must not be set to UNLIM. A value of 0 (zero) is invalid.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	In *non-conversational* mode, message containers are released as soon as the client receives a reply from the server. If no reply is requested, message containers are released as soon as the server receives the client request. In *conversational* mode, the last message received is always kept until a new one is received. **Note:** 1. If a catch-all service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 2. See ***Wildcard Service Definitions***.						
NUM-PARTICIPANT-EXTENSION	*n*	O	z	u	w	v	b
	Defines the number of participant extensions to link participants as clients and servers. *n* Number of participant extensions. *not specified* If this attribute is not set, the default value is calculated based on NUM-CLIENT and NUM-SERVER. A value of 0 (zero) is invalid.						
NUM-SERVER	*n* | AUTO	R	z	u	w	v	b
	Defines the number of servers that can offer services concurrently using the broker. This is *not* the number of services that can be registered to the broker (see NUM-SERVICE). *n* Number of servers. AUTO Uses the SERVER-DEFAULT and the service-specific SERVER-LIMIT values to calculate the number of servers. 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 ***Wildcard Service Definitions***.						
NUM-SERVICE	*n*	R	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	Defines the number of services that can be registered to the broker. This is *not* the number of servers that can offer the services (see NUM-SERVER). A value of 0 (zero) is invalid.						
NUM-SERVICE-EXTENSION	*n* | AUTO	O	z	u	w	v	b
	Defines the number of service extensions to link servers to services. *n* Number of service extensions. AUTO Uses the value specified or calculated for NUM-SERVER + NUM-CLIENT, plus an extra cushion. *not specified* If this attribute is not set, the default value is NUM-SERVER multiplied by NUM-SERVICE. The minimum value is NUM-SERVER. The maximum value is NUM-SERVER multiplied by NUM-SERVICE. Caution is recommended with this attribute: ■ Set this attribute only if the storage resources allocated for service extensions need to be restricted. ■ Note that the value *n* allows only the specified number of server instances of *n* to be used. ■ Value AUTO will calculate the number of allowed server instances from NUM-SERVER, which itself might be set to AUTO. In this case, this also considers the value of SERVER-DEFAULT and even the individual SERVER-LIMIT for each service definition.						
NUM-SHORT-BUFFER or NUM-SHORT	*n* | AUTO	R	z	u	w	v	b
	Defines the number of short message containers. Short message containers have a fixed length of 256 bytes and are used to store requests of no more than 2048 bytes. To store a request of 1024 bytes, for example, would require four short message containers. *n* Number of buffers. AUTO Uses the SHORT-BUFFER-DEFAULT and the service-specific SHORT-BUFFER-LIMIT values to calculate the number of short message buffers. The values used in the calculation must not be set to UNLIM. **Note:** 1. In *non-conversational* mode, message containers are released as soon as the client receives a reply from the server. If no reply is requested, message containers are released as soon as the server receives the client request.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	2. In *conversational* mode, the last message received is always kept until a new one is received. 3. If a wildcard service is defined in the service-specific section of the attribute file, the value of AUTO is invalid. 4. See ***Wildcard Service Definitions***.						
NUM-UOW	0 | *n*	O	z	u	w	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 NUM-UOW value must be 1 or larger for the broker. (MAX-UOWS is an alias for this attribute.) The NUM-UOW value for the service will default to the value set for the broker.						
NUM-WORKER	1 | *n* (max. 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	YES | 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 *Protecting a Broker against Denial-of-Service Attacks* 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 *Transport-method-style Broker ID* for more details. This attribute is required if the attribute RUN-MODE is specified.						
PERCENTAGE-FOR-CONNECTION-	90 | 1-100	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
SHORTAGE-MESSAGE	Broker will issue a message if the defined percentage value of TCP/IP connections (available file descriptors) is exceeded. Default is 90 percent of the available file descriptors.						
POLL	YES | NO	O	z	u		v	
	In earlier EntireX versions, the maximum number of TCP/IP connections per communicator was limited; see *Maximum TCP/IP Connections per Communicator* under *Broker Resource Allocation* for platform-specific list. With attribute POLL introduced in EntireX version 9.0, this restriction can be lifted under z/OS, UNIX and z/VSE. NO This setting is used to run the compatibility mode in Broker. The poll() system call is not used. The limitations described under *Maximum TCP/IP Connections per Communicator* under *Broker Resource Allocation* apply. YES The poll() system call is used to lift the resource restrictions with select() in multiplexing file descriptor sets. **Note:** The maximum number of file descriptors per process is a hard limit that cannot be exceeded by POLL=YES. Setting this attribute to YES increases CPU consumption. POLL=YES is only useful if ■ you need more than the maximum number of TCP/IP connections per communicator, as described under *Maximum TCP/IP Connections per Communicator* under *Broker Resource Allocation*, and ■ this maximum number is less than the maximum number of file descriptors per process We recommend POLL=NO to reduce CPU consumption.						
PSTORE	NO | HOT | COLD	O	z	u	w	v	b
	Defines the status of the persistent store at broker startup, including the condition of persistent units of work (UOWs). With any value other than NO, PSTORE-TYPE must be set. NO No persistent store. HOT Persistent UOWs are restored to their prior state during initialization. COLD Persistent UOWs are not restored during initialization, and the persistent store is considered empty. **Note:** For a hot or cold start, the persistent store must be available when your broker is restarted.						
PSTORE-REPORT	NO | YES	O	z	u	w	v	b
	Determines whether PSTORE report is created.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	`NO` Do not create the PSTORE report file. `YES` Create the PSTORE report file. See also *Persistent Store Report*.						
`PSTORE-TYPE`	`DIV` (z/OS) | `CTREE` (UNIX, Windows) | `ADABAS` (all platforms) | `FILE` (UNIX, Windows)	O	z	u	w	v	b
	Describes the type of persistent store driver required. `DIV` Data in Virtual. z/OS only, and default on this platform. See ***DIV-specific Attributes*** below and *Implementing a DIV Persistent Store* under *Managing the Broker Persistent Store*. `CTREE` c-tree database. UNIX and Windows only. See ***c-tree-specific Attributes*** and *c-tree Database as Persistent Store* in the UNIX and Windows Administration documentation. `ADABAS` Adabas. All platforms. See also ***Adabas-specific Attributes*** (below) and *Managing the Broker Persistent Store* 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 *Unique Message ID*. `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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
RUN-MODE	STANDARD | STANDBY | PSTORE-LOAD | PSTORE-UNLOAD	O	z	u	w	v	b
	Determines the initial run mode of the broker. STANDARD Default value. Normal mode. STANDBY Deprecated. Supported for compatibility reasons. PSTORE-LOAD Broker will run as load broker to write Persistent Store data to a new persistent store. See also *Migrating the Persistent Store*. 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 *Migrating the Persistent Store*.						
SECURITY	NO | YES	O	z	u	w	v	b
	Determines whether EntireX Security is activated. NO EntireX Security is not activated. YES EntireX Security is activated. See *EntireX Security*.						
SERVER-DEFAULT	*n* | UNLIM	O	z	u	w	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 *without* a restart. The attribute file is read only when the first server registers for a particular service; it is not reread when a second replica is activated. NO The attribute file is read only once during broker startup. Any changes to the attribute file will be honored only if the broker is restarted.						
SHORT-BUFFER-DEFAULT	UNLIM | *n*	O	z	u	w	v	b
	Number of short buffers to be allocated for each service.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	UNLIM The number of short message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM-SHORT-BUFFER=AUTO. *n* Number of buffers. This value can be overridden by specifying a SHORT-BUFFER-LIMIT for the service. A value of 0 (zero) is invalid.						
STORAGE-REPORT	NO | YES	O	z	u	w	v	b
	Create a storage report about broker memory usage. NO Do not create the storage report. YES Create the storage report. See *Storage Report*.						
STORE	OFF | BROKER	O	z	u	w	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 *Flushing Trace Data to a GDG Data Set* under *Tracing EntireX Broker*. The following keywords are supported as part of the TRACE-DD value: ■ DATACLAS ■ DCB including BLKSIZE, DSORG, LRECL, RECFM ■ DISP ■ DSN ■ MGMTCLAS ■ SPACE ■ STORCLAS ■ UNIT Refer to your JCL Reference Manual for a complete description of the syntax. Example:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	`TRACE-DD = "DSNAME=EXX.GDG, DCB=(BLKSIZE=1210,DSORG=PS,LRECL=121,RECFM=FB), DISP=(NEW,CATLG,CATLG), SPACE=(CYL,(100,10)), STORCLAS=SMS"` **Note:** If you specify `TRACE-DD`, you must also specify `TRMODE=WRAP` and a value for `TRBUFNUM` for the setting to take effect.						
`TRACE-FILE-SIZE`	*n* | *n*`K` | *n*`M` | *n*`G`	O		u	w		
	Defines the size of one trace file in kilobytes, megabytes or gigabytes. If this size is exceeded, a new trace file is allocated until the maximum number of trace files specified with `MAX-TRACE-FILES` is reached. There is no default value. These two parameters help prevent a constantly growing ETB.LOG file. See *Trace File Handling* 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.						

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

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	communications with the PSTORE. Broker will reject any user requests until it is able to contact the Adabas database. If broker should start with a PSTORE-TYPE=CTREE and the c-tree files are still in use, broker will retry every 10 seconds to rebuild the persistent data. Broker will reject any user requests until it is able to rebuild the persistent data.						
WORKER-MAX	32 | *n* (min. 1, max. 32)	O	z	u	w		b
	Maximum number of worker tasks the broker can use.						
WORKER-MIN	1 | *n* (min. 1, max. 32)	O	z	u	w		b
	Minimum number of worker tasks the broker can use.						
WORKER-NONACT	70S | *n* | *n*S | *n*M | *n*H	O	z	u	w		b
	Non-activity time to elapse before a worker tasks is stopped. *n* Same as *n*S. *n*S Non-activity time in seconds (default 70, max. 2147483647). *n*M Non-activity time in in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). **Caution:** A value of 0 (zero) is invalid. If you set this value too low, additional overhead is required for starting and stopping worker tasks. The default and recommended value is 70S.						
WORKER-QUEUE-DEPTH	1 | *n* (min. 1)	O	z	u	w		b
	Number of unassigned user requests in the input queue before another worker task gets started. The default and recommended value is 1. A higher value will result in longer broker response times.						
WORKER-START-DELAY	*internal-value* | *n*	O	z	u	w		b
	n Delay is extended by *n* seconds. Delay after a successful worker task invocation before another worker task can be started to handle current incoming workload. This attribute is used to avoid the risk of recursive invocation of worker tasks, because starting a worker task itself causes workload increase. If no value is specified, an internal value calculated by the broker is used to optimize dynamic worker management. This calculated value is the maximum time required to start a worker task.						

Service-specific Attributes

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
APPLICATION-MONITORING or APPMON	YES | NO	O	z	u	w	v	b
	YES Enable application monitoring for the specified services. NO Disable application monitoring for the specified services. See *Application Monitoring*.						
APPLICATION-MONITORING-NAME or APPMON-NAME	A100	O	z	u	w	v	b
	Specifies the application monitoring name. Used to set the value of the **ApplicationName** KPI. If omitted, the default value from the APPLICATION-MONITORING section is used. If this value is also not specified, the corresponding CLASS/SERVER/SERVICE names are used. See *Application Monitoring*.						
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	N | Y	O	z				b
	Determines whether this service is subject to RPC authorization checking. N No RPC authorization checking is performed. Y RPC library and program name are appended to the authorization check performed by EntireX Security. Specify YES only to RPC-supported services.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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>).						
CONV-LIMIT	UNLIM | *n*	O	z	u	w	v	b
	Allocates a number of conversations especially for this service. 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. *n* Number of conversations. A value of 0 (zero) is invalid. If NUM-CONVERSATION=AUTO is specified in the Broker section of the attribute file, CONV-LIMIT=UNLIM is not allowed in the service section. A value must be specified or the CONV-LIMIT attribute must be suppressed entirely for the service so that the default (CONV-DEFAULT) becomes active.						
CONV-NONACT	5M | *n* | *n*S | *n*M | *n*H	R	z	u	w	v	b
	Non-activity time for connections. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647). *n*M Non-activity time in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). A value of 0 (zero) is invalid. If a connection is not used for the specified time, that is, a 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.						
CONVERSION	A255 (SAGTCHA[,TRACE=*n*][,*OPTION*=*s*]| SAGTRPC[,TRACE=*n*][,*OPTION*=*s*]| *name*[,TRACE=*n*]| NO)	O	z	u	w	v	b
	Defines ICU conversion or SAGTRPC user exit for character conversion. See *Internationalization with EntireX*. SAGTCHA (1) Conversion using ICU Conversion for *ACI-based Programming*. SAGTRPC (2) Conversion using ICU Conversion for *RPC-based Components* and *Reliable RPC*. *name* (3) Name of the SAGTRPC user exit for RPC-based components and Reliable RPC. See also *Configuring SAGTRPC User Exits* under						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000

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

`DEFERRED`	`NO` | `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`	`YES` | `NO`	O	z	u	w	v	b

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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	UNLIM | *n*	O	z	u	w	v	b
	Allocates a number of long message buffers for the service. UNLIM The number of long message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM-LONG-BUFFER=AUTO in the Broker section of the attribute file. *n* Number of long message buffers. A value of 0 (zero) is invalid. If NUM-LONG-BUFFER=AUTO is specified in the Broker 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.						
MAX-MESSAGES-IN-UOW	16 | *n*	O	z	u	w	v	b
	Maximum number of messages in a UOW.						
MAX-MESSAGE-LENGTH	2147483647 | *n*	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 | *n*	O	z	u	w	v	b
	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. *n* 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. Specify MAX-UOWS=0 for Natural RPC Servers. This restriction will be removed with a later release.						
MUOW	See MAX-UOWS.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
NOTIFY-EOC	NO | YES	O	z	u	w	v	b
	Specifies whether timed-out conversations are to be stored or discarded. NO Discard the EOC notifications if the server is not ready to receive. YES Store the EOC notifications if the server is not ready to receive and then notify the server if possible. If a server is not ready to receive an EOC notification, it can be stored or discarded. If it is stored, the server is notified, if possible, when it is ready to receive. **Caution:** The behavior activated by this parameter can be relied upon only during a single lifetime of the broker kernel. Specifically, conversations containing units of work, whose lifetime can span multiple broker kernel sessions, cannot be assumed to show this behavior, even with NOTIFY-EOC=YES.						
NUM-UOW	Alias for MAX-UOWS.						
POSTPONE-ATTEMPTS	0 | *n*	O	z	u	w		
	Defines the number of attempts putting a received unit of work (UOW) due to SYNCPOINT option CANCEL on the postpone queue for later processing. 0 All UOWs rejected by the receiver (SYNCPOINT option CANCEL) will be cancelled immediately. Attribute POSTPONE-DELAY is ignored. *n* Defines the number of postpone attempts that are performed instead of considering the UOW finished due to SYNCPOINT option CANCEL; the UOW will be moved to the 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. The default value is 0. See *Postponing Units of Work*.						
POSTPONE-DELAY	0 | *n* | *n*S | *n*M | *n*H	O	z	u	w		
	The length of time a UOW is kept in status POSTPONED. 0 The postpone feature is disabled. Attribute POSTPONE-ATTEMPTS is ignored. *n*S Number of seconds the UOW stays unreadable in the postpone queue with status POSTPONED (max. 2147483647). *n*M Number of minutes the UOW stays unreadable in the postpone queue with status POSTPONED (max. 35791394). *n*H Number of hours the UOW stays unreadable in the postpone queue with status POSTPONED (max. 596523). *n*D Number of days the UOW stays unreadable in the postpone queue with status POSTPONED (max. 24855).						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	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. **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.						
SERVER	A32 (case-sensitive)	R	z	u	w	v	b
	Part of the name that identifies the service together with the CLASS and SERVICE attributes. CLASS must be specified first, followed immediately by SERVER and SERVICE. Valid characters for server name are letters a-z, A-Z, numbers 0-9, hyphen and underscore. Do not use dollar, percent, period or comma.						
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. A value of 0 (zero) is invalid. This value can be overridden by specifying a SERVER-LIMIT for the service.						
SERVER-LIMIT	*n* | UNLIM	O	z	u	w	v	b
	Allows a number of servers especially for this service. *n* Number of servers. UNLIM The number of servers is restricted only by the number of servers globally available. Precludes the use of NUM-SERVER=AUTO in the Broker section of the attribute file. A value of 0 (zero) is invalid. 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. **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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
SERVER-NONACT	5M | *n* | *n*S | *n*M | *n*H	R	z	u	w	v	b
	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. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647). *n*M Non-activity time in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). If a server registers multiple services, the highest value of all the services registered is taken as non-activity time for the server.						
SERVICE	A32 (case-sensitive)	R	z	u	w	v	b
	Part of the name that identifies the service together with the CLASS and SERVER attributes. CLASS must be specified first, followed immediately by SERVER and SERVICE. The SERVICE attribute names EXTRACTOR and DEPLOYMENT are reserved for Software AG internal use and should not be used in customer-written applications. Valid characters for service name are letters a-z, A-Z, numbers 0-9, hyphen and underscore. Do not use dollar, percent, period or comma. See also the restriction for CLASS attribute names.						
SHORT-BUFFER-LIMIT	UNLIM | *n*	O	z	u	w	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. *n* Number of short message buffers. If NUM-SHORT-BUFFER=AUTO is specified in the Broker section of the attribute file, SHORT-BUFFER-LIMIT=UNLIM is not allowed in the service section. A value must be specified or the SHORT-BUFFER-LIMIT attribute must be suppressed entirely for the service so that the default (SHORT-BUFFER-DEFAULT) becomes active.						
STORE	OFF | BROKER	O	z	u	w	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.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	This attribute can be overridden by the STORE field in the Broker ACI control block.						
TRANSLATION	NO | *name* (A255)	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. *name* Name of Translation User Exit. See also *Configuring Translation User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation or *Writing Translation User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. The CONVERSION attribute overrides the TRANSLATION attribute when defined for a service; that is, when TRANSLATION and CONVERSION are both defined, TRANSLATION will be ignored.						
UMSG	Alias for MAX-MESSAGES-IN-UOW.						
UOW-DATA-LIFETIME	1D | *n*S | *n*M | *n*H | *n*D	O	z	u	w	v	b
	Defines the default lifetime for units of work for the service. *n*S Number of seconds the UOW can exist (max. 2147483647). *n*M Number of minutes the UOW can exist (max. 35791394). *n*H Number of hours the UOW can exist (max. 596523). *n*D Number of days the UOW can exist (max. 24855). This attribute is ignored if PSTORE=NO is defined. If the unit of work (UOW) is inactive, that is, not processed within the time limit, it is deleted and given a status of TIMEOUT. This attribute can be overridden by the UWTIME field in the Broker ACI control block.						
UOW-MSGS	Alias for MAX-MESSAGES-IN-UOW.						
UOW-STATUS-LIFETIME	no value | *n*[S] | *n*M | *n*H | *n*D	O	z	u	w	v	b
	The value to be added to the UOW-DATA-LIFETIME (lifetime of associated UOW). If a value is entered, it must be 1 or greater; a value of 0 will result in an error. If no value is entered, the lifetime of the UOW *status* information will be the same as the lifetime of the UOW itself. *n*S Number of seconds the UOW status exists longer than the UOW itself (max. 2147483647). *n*M Number of minutes (max. 35791394). *n*H Number of hours (max. 596523).						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	*n*D Number of days (max. 24855). The lifetime determines how much additional time the UOW status is retained in the persistent store and is calculated from the time at which the associated UOW enters any of the following statuses: PROCESSED, TIMEOUT, BACKEDOUT, CANCELLED, DISCARDED. The additional lifetime of the UOW status is calculated only when broker is executing. Value in UOW-STATUS-LIFETIME supersedes the value (if specified) in attribute UWSTATP. **Note:** If no unit is specified, the default unit is seconds. The unit does not have to be identical to the unit specified for UOW-DATA-LIFETIME.							
UWSTATP	0 | *n*	O	z	u	w	v	b
	Contains a multiplier used to compute the lifetime of a persistent status for the service. The UWSTATP value is multiplied by the UOW-STATUS-LIFETIME value (the lifetime of the associated UOW) to determine the length of time the status will be retained in the persistent store. 0 The status is not persistent. 1 - 254 Multiplied by the value of UOW-DATA-LIFETIME to determine how long a persistent status will be retained. This attribute is ignored if PSTORE=NO is defined. **Note:** This attribute has not been supported since EntireX version 7.3. Use UOW-STATUS-LIFETIME instead.							
UWSTAT-LIFETIME	Alias for UOW-STATUS-LIFETIME.						
UWTIME	Alias for UOW-DATA-LIFETIME.						

Wildcard Service Definitions

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

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

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

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

Service Update Modes

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

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

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

OPTION Values for Conversion

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

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

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

Value	Description	Options Supported for		Report Situation in Broker Log File if TRACE Option for CONVERSION is set to 1	
		SAGTCHA	SAGTRPC	Bad Input Characters (Sender's Codepage)	Non-convertible Characters (Receiver's Codepage)
SUBSTITUTE	Substitutes both non-convertible characters (receiver's codepage) and bad input characters (sender's codepage) with a 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

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

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000

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

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

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

`locale-string`	Any ICU converter name or alias. See also ***Additional Notes*** below.	O	z	u	w	v	
	Customize the mapping of locale strings to codepages and bypass the broker's locale string processing mechanism. See *Broker's Locale String Processing*. This is useful: ■ if the broker's locale string processing fails - 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 *Building and Installing ICU Custom Converters* in the platform-specific Administration documentation. The attribute (locale string) is the locale string sent by your EntireX component (client or server) and the value is the codepage that you want to use in place of that locale string. In the first line of the example below, the client or server application sends ASCII as a locale string; the broker maps this to the codepage ISO 8859_1. In the same way EUC_JP_LINUX is mapped to ibm-33722_P12A-1999. All other locale strings are mapped by the broker's mapping mechanism, see *Broker's Built-in Locale String Mapping*. Example:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	DEFAULTS=CODEPAGE * Broker Locale String Codepage Assignments ASCII=ISO8859 EUC_JP_LINUX=ibm-33722_P12A-1999 * Customer-written ICU converters CP1140=myebcdic CP0819=myascii For more examples, see *Bypassing Broker's Built-in Locale String Mapping* and also ***Additional Notes*** below.						

Additional Notes

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

Adabas SVC/Entire Net-Work-specific Attributes

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

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

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

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

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

Security-specific Attributes

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

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

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

AUTHORIZATION-DEFAULT	YES | NO	O		u	w		
	Determines whether access is granted to a specified service if the specified service could not be found listed in the repository of authorization rules or in section *`DEFAULTS=AUTHORIZATION-RULES`* of the attribute file. YES Grant access. NO Deny access. Applies only when using EntireX Security under 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. See also *Authorization Rules*.						
CHECK-IP-ADDRESS	YES | NO	O	z				
	Determines whether the TCP/IP address of the caller is subject to a resource check.						

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

	Authentication is performed against the LDAP repository specified under *ldapUrl*.
	
	■ **TCP**
	Specify repository URL:
	
	`LDAP-AUTHENTICATION-URL="ldap://HostName[:PortNumber]"`
	
	■ **SSL/TLS**
	Specify repository URL with ldaps:
	
	`LDAP-AUTHENTICATION-URL="ldaps://HostName[:PortNumber]"`
	
	If no port number is specified, the default is the standard LDAP port number 389 for TCP transport. Examples for TCP and SSL/TLS:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	`LDAP-AUTHENTICATION-URL="ldap://myhost.mydomain.com"` `LDAP-AUTHENTICATION-URL="ldaps://myhost.mydomain.com:636"`						
`LDAP-AUTHORIZATION-URL`	*`ldapUrl`*	O		u	w		

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

`LDAP-AUTH-DN`	*`authDN`*	O		u	w		
	For authenticated access to the LDAP server. Specifies the DN of the user. Default value: `cn=admin,dc=software-ag,dc=de` This attribute replaces parameter `authDN` in the *xds.ini* file of EntireX version 9.10 and below.						
`LDAP-AUTH-PASSWD-ENCRYPTED`	*`authPass`*	O		u	w		
	For authenticated access to the LDAP server. Specifies the encrypted value of the user password. Use program `etbnattr` to get the encrypted password: `etbnattr -x clear_text_password -echo_password_only` This writes the encrypted password to standard output. This attribute replaces parameter `authPass` in the *xds.ini* file of EntireX version 9.10 and below.						
`LDAP-AUTHORIZATION-RULE`	`A32`	O		u	w		
	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 `SECURITY-SYSTEM=`*`ldapUrl`*. Authorization rules can be stored in an LDAP repository. When an authorization call occurs, EntireX Security uses the values of this parameter						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	and AUTHORIZATION-DEFAULT to perform an access check for a particular broker instance against an (authenticated) user ID and list of rules. See also *Authorization Rules*.						
LDAP-BASE-DN	*baseDN*	O		u	w		
	Specifies the base distinguished name of the directory object that is the root of all objects for authorization rules. Default value: `dc=software-ag,dc=de` This attribute replaces parameter baseDN in the *xds.ini* file of EntireX version 9.10 and below.						
LDAP-PERSON-BASE-BINDDN	*ldapDn*	O		u	w		
	Used with LDAP authentication to specify the distinguished name where authentication information is stored. This value is prefixed with the user ID field name (see below). Example: `LDAP-PERSON-BASE-BINDDN="cn=users,dc=mydomain,dc=com"`						
LDAP-REPOSITORY-TYPE	OpenLDAP | ActiveDirectory | SunOneDirectory | Tivoli | Novell | ApacheDS	O		u	w		
	Use predefined known fields for the respective repository type. Specify the repository type that most closely matches your actual repository. In the case of Windows Active Directory, the user ID is typically in the form *domainName**userId*.						
LDAP-SASL-AUTHENTICATION	NO | YES	O			w		
	Specifies whether or not Simple Authentication and Security Layer (SASL) is to perform the authentication check. In practice, this determines whether or not the password supplied by the user is passed in plain text between the broker kernel and the LDAP server. If SASL is activated, this implies that the password is encrypted. NO Password is sent to LDAP server in plain text. YES Password is sent to LDAP server encrypted.						
LDAP-USERID-FIELD	cn | *uidFieldName*	O		u	w		
	Used with LDAP authentication to specify the first field name of a user in the Distinguished Name, for example: `LDAP-USERID-FIELD=uid`						
MAX-SAF-PROF-LENGTH	1-256	O	z				
	This parameter should be increased if the length of the resource checks - that is, the length of the profile comprising “<class>.<server>.<service>” - is greater than 80 bytes.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	This parameter defaults to 80 if a value is not specified.						
PASSWORD-TO-UPPER-CASE	NO | YES	O	z			v	
	Determines whether the password and new password are converted to uppercase before verification.						
PRODUCT	RACF | ACF2 | TOP-SECRET	O	z				
	Specifies the name of the installed security product. This attribute is used to analyze security-system-specific errors. The following systems are currently supported: 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	YES | NO	O	z				
	Determines whether a client user ID obtained by means of the trusted user ID mechanism is propagated to a server using the ACI field CLIENT-USERID.						
SAF-CLASS	NBKSAG | *SAFClassName*	O	z				
	Specifies the name of the SAF class/type used to hold the EntireX-related resource profiles.						
SAF-CLASS-IP	NBKSAG | *SAFClassName*	O	z				
	Specifies the name of the SAF class/type used when performing IP address authorization checks.						
SECURITY-LEVEL	AUTHORIZATION | AUTHENTICATION	O	z	u	w	v	b
	Specifies the mode of operation. AUTHORIZATION Authorization and authentication (not under BS2000 or z/VSE). AUTHENTICATION Authentication. **Note:** In version 8.0, the default value for this parameter was AUTHORIZATION.						
SECURITY-NODE	YES | *name*	O	z				
	This parameter can be used to specify a prefix that is added to all authorization checks, enabling different broker kernels, in different environments, to perform separate authorization checks according to each broker kernel. For example, it is often important to distinguish between production, test, and development environments. YES This causes the broker ID to be used as a prefix for all authorization checks.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	name This causes the actual text (maximum 8 characters) to be prefixed onto all authorization checks. **Note:** By *not* setting this parameter, no prefix is added to the resource check (the default behavior).						
SECURITY-SYSTEM	OS | LDAP	O	z	u	w		b
	OS Authentication is performed against the local operating system. Default if SECURITY=YES is specified and section DEFAULTS=SECURITY is omitted from the attribute file. LDAP Authentication and authorization are performed against the LDAP repository specified under LDAP-AUTHENTICATION-URL and LDAP-AUTHORIZATION-URL.						
TRACE-LEVEL	0-4	O	z	u	w	v	b
	Trace level for EntireX Security. It overrides the global value of trace level in the attribute file. 0 No tracing. Default value. 1 Log security violations and access denied/permitted. 2 All of trace level 1, plus internal errors. 3 All of trace level 2, plus function entered/exit messages with argument values and some progress messages. 4 All of trace level 3, plus some selected data areas for problem analysis. Trace levels 2, 3 and 4 should be used only when requested by Software AG support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD. **Note:** Setting this value also affects tracing for authorization rules.						
TRUSTED-USERID	YES | NO	O	z				
	Activates the trusted user ID mechanism for broker requests arriving over the local Adabas IPC mechanism.						
USERID-TO-UPPER-CASE	NO | YES	O	z			v	
	Determines whether user ID is converted to uppercase before verification.						
UNIVERSAL	NO | YES	O	z				
	Determines whether access to undefined resource profiles is allowed.						
WARN-MODE	NO | YES	O	z	u	w		b
	Determines whether a resource check failure results in just a warning or an error.						

TCP/IP-specific Attributes

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
CONNECTION-NONACT	*n* | *n*S | *n*M | *n*H	O	z	u	w	v	b
	Non-activity of the TCP/IP connection, after which a close is performed and the connection resources are freed. If this parameter is not specified here, broker will close the connection only when the application (or the network itself) terminates the connection. *n* Same as *n*S. *n*S Non-activity time in seconds (min. 600, max. 2147483647). *n*M Non-activity time in minutes (min. 10, max. 35791394). *n*H Non-activity time in hours (max. 596523). If not specified, the connection non-activity test is disabled. On the stub side, non-activity can be set with the environment variable ETB_NONACT. See *Limiting the TCP/IP Connection Lifetime* in the platform-specific *Stub Administration* sections of the EntireX documentation.						
HOST	0.0.0.0 | *HostName* | *IP address*	O	z	u	w	v	b
	The address of the network interface on which broker will listen for connection requests. If HOST is not specified, broker will listen on any attached interface adapter of the system (or stack). A maximum of five HOST/PORT pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.						
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	v	b
	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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000

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

`RESTART`	`YES` | `NO`	O	z	u	w	v	b
	`YES` The broker kernel will attempt to restart the TCP/IP communicator. `NO` The broker kernel will not try to restart the TCP/IP communicator. This setting applies to all TCP/IP communicators.						
`RETRY-LIMIT`	`20` | *n* | `UNLIM`	O	z	u	w	v	b
	Maximum number of attempts to restart the TCP/IP communicator. This setting applies to all TCP/IP communicators.						
`RETRY-TIME`	`3M` | *n* | *n*`S` | *n*`M` | *n*`H`	O	z	u	w	v	b
	Wait time between stopping the TCP/IP communicator due to an unrecoverable error and the next attempt to restart it. *n* Same as *n*`S`. *n*`S` Wait time in seconds (max. 2147483647). *n*`M` Wait time in minutes (max. 35791394). *n*`H` Wait time in hours (max. 596523). Minimum wait time is `1S`.						

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

c-tree-specific Attributes

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

Not available under z/OS, BS2000, z/VSE.

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

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

SSL/TLS-specific Attributes

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

Your operating system and, for z/OS, the approach you use determine whether this section of the attribute file is required:

- **z/OS**
  - **AT-TLS**
    This is the approach we recommend. IBM's Application Transparent Transport Layer Security (AT-TLS) does not require the SSL-specific attribute section.
  - **Direct SSL**
    For direct SSL/TLS support, the SSL-specific attribute section is required. It begins with the keyword `DEFAULTS=SSL` as shown in the sample attribute file.

    Note: Direct SSL/TLS support (using GSK) inside the Broker under z/OS will be dropped in the next version. We strongly recommend using IBM's Application Transparent Transport Layer Security (AT-TLS) instead.

  See *Running Broker with SSL/TLS Transport* in the z/OS Administration documentation.
- **UNIX and Windows**
  The SSL-specific attribute section is required, and begins with the keyword `DEFAULTS=SSL` as shown in the sample attribute file.

  The attributes in this section are needed to execute the SSL communicator of the EntireX Broker kernel.

  See *Running Broker with SSL/TLS Transport*.
- **z/VSE**
  The SSL-specific attribute section is not used. You can use BSI's Automatic Transport Layer Security (ATLS). See *Running Broker with SSL/TLS Transport* in the z/VSE Administration documentation.

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
CIPHER-SUITE	*string*	O	z	u	w		b
	String that is passed to the underlying SSL/TLS implementation. SSL/TLS is a standardized protocol that uses different cryptographic functions (hash functions, symmetric and asymmetric encryption etc.). Some of these must be implemented in the SSL/TLS stack; others are optional. When an SSL/TLS connection is created, both parties agree by "handshake" on the cipher suite, that is, the algorithms and key lengths used. In a default scenario, this information depends on what both sides are capable of. It can be influenced by setting the attribute CIPHER-SUITE for the SSL/TLS server side (the broker always implements the server side). Thus stubs connect to the broker and thereby become the SSL/TLS clients. Under UNIX, Windows and BS2000, the OpenSSL implementation is used; under z/OS it is GSK. The SSL protocol is obsolete. It is no longer available. The TLS protocol is the successor of SSL and is readily available in OpenSSL and GSK. The following examples show how to configure the available cipher suites: ■ **OpenSSL** The default configuration uses FIPS 140-2 approved cipher suites, eligible for TLS v1.2, but without anonymous Diffie-Hellman (ADH) and pre-shared key (PSK) algorithms. The resulting set of cipher suites provides for authentication and strong encryption: CIPHER-SUITE=FIPS+TLSv1.2:!ADH:!PSK:@STRENGTH See ***https://www.openssl.org/docs/man1.1.1/man1/ciphers***. ■ **GSK** Default configuration: CIPHER-SUITE=9F9D9E9C6B673D3C39383332352F This list of FIPS 140-2 approved cipher suites starts with a strong '256-bit AES in Galois Counter Mode encryption with 128-bit AEAD authentication and ephemeral Diffie-Hellman key exchange signed with an RSA certificate' (9F) and ends with a relatively weak '128-bit AES encryption with SHA-1 message authentication and RSA key exchange' (2F). See the IBM documentation *z/OS V2R3 Cryptographic Services System Secure Sockets Layer Programming*, SC14-7495-30, Appendix C: *Cipher Suite Definitions*.						
CONNECTION-NONACT	*n* | *n*S | *n*M | *n*H	O	z	u	w		b
	Non-activity of the SSL connection, after which a close is performed and the connection resources are freed. If this parameter is not specified here, broker will close the connection only when the application (or the network itself) terminates the connection.						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	n Same as *n*S. *n*S Non-activity time in seconds (min. 600, max. 2147483647). *n*M Non-activity time in minutes (min. 10, max. 35791394). *n*H Non-activity time in hours (max. 596523). If not specified, the connection non-activity test is disabled.						
HOST	*hostname*	O	z	u	w		b
	The address of the network interface on which broker will listen for connection requests. If HOST is not specified, broker will listen on any attached interface adapter of the system (or stack). A maximum of five HOST/PORT pairs can be specified to start multiple instances of EntireX Broker's TCP/IP transport communicator.						
KEY-LABEL	*name*	O	z				
	The label of the key in the RACF keyring that is used to authenticate the broker kernel (see also TRUST-STORE parameter). Example: ETBCERT.						
KEY-FILE	*filename*	R		u	w		b
	File that contains the broker's private key (if not contained in KEY-STORE). For test purposes, EntireX delivers certificates for use on various platforms. See *SSL/TLS Sample Certificates Delivered with EntireX*. Example for UNIX and Windows: MyAppKey.pem. **Note:** EntireX Broker does not support Java certificates (keystore files of type .jks).						
KEY-PASSWD	*password* (A32)	R		u	w		b
	Password used to protect the private key. Unlocks the KEY-FILE, for example MyAppKey.pem. Deprecated. See KEY-PASSWD-ENCRYTPED below.						
KEY-PASSWD-ENCRYPTED	*encrypted value* (A64)	R		u	w		b
	Password used to protect the private key. Unlocks the KEY-FILE, for example MyAppKey.pem. This attribute replaces KEY-PASSWD to avoid a clear-text password as attribute value. If KEY-PASSWD and KEY-PASSWD-ENCRYTPED are both supplied, KEY-PASSWD-ENCRYTPED takes precedence. Use program etbnattr to get the encrypted password:						

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	`etbnattr -w ssl_key_password --echo_password_only` This writes the encrypted password to standard output.						
KEY-STORE	*file name*	R		u	w		b
	SSL certificate; may contain the private key. For test purposes, EntireX delivers certificates for use on various platforms. See *SSL/TLS Sample Certificates Delivered with EntireX*. Example for UNIX and Windows: *ExxAppCert.pem*. **Note:** EntireX Broker does not support Java certificates (keystore files of type .jks).						
MAX-MESSAGE-LENGTH	2147483647 | *n*	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	1025-65535	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	YES | 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	20 | *n* | UNLIM	O	z	u	w		b
	Maximum number of attempts to restart the SSL communicator.						
RETRY-TIME	3M | *n* | *n*S | *n*M | *n*H	O	z	u	w		b
	Wait time between suspending SSL communication due to unrecoverable error and the next attempt to restart it. *n* Same as *n*S. *n*S Wait time in seconds (max.2147483647). *n*M Wait time in minutes (max. 35791394). *n*H Wait time in hours (max. 596523). Minimum: 1S						
REUSE-ADDRESS	YES | NO	O	z	u	w		b

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	UNIX	Windows	z/VSE	BS2000
	■ **UNIX/Windows/BS2000** Specify the file name of the CA certificate store. Examples: `EXXCACERT.PEM`, `C:\Certs\ExxCACert.pem`						
`VERIFY-CLIENT`	`NO` | `YES`	O	z	u	w		b
	`YES` Additional client certificate required. `NO` No client certificate required (default).						

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

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

Adabas-specific Attributes

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

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

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

Application Monitoring-specific Attributes

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

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

Authorization Rule-specific Attributes

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

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

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

Attribute	Values	Opt/ Req	Operating System				
			z/OS	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 *Rules Stored in Broker Attribute File*.						

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

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

Variable Definition File

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

The following attributes are considered unique for each machine:

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

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

5 Configuring Broker for Internationalization

Software internationalization is the process of designing products and services so that they can be adapted easily to a variety of different local languages and cultures. Internationalization within EntireX means internationalization of messages: the incoming and outgoing messages are converted to the desired codepage of the platform in use. This chapter explains in detail how to configure the broker for character conversion.

See also *Internationalization with EntireX*.

Configuring ICU Conversion

To configure ICU conversion

1 In the Broker attribute file, set the service-specific attribute `CONVERSION`. Examples:

- ICU Conversion with SAGTCHA for *ACI-based Programming*:

```
CONVERSION=(SAGTCHA,OPTION=SUBSTITUTE)
```

- ICU Conversion with SAGTRPC for *RPC-based Components* and *Reliable RPC*:

```
CONVERSION=(SAGTRPC,OPTION=STOP)
```

2 Optionally configure a `CONVERSION OPTION` to tune error behavior to meet your requirements; see *`OPTION` Values for Conversion*.

3 For the Broker attribute, check if ICU conversion is possible, that is, the attribute `ICU-CONVERSION` is either

- not defined, its default is `YES`
- set to `YES`

To configure locale string defaults (optional)

- If the broker's locale string defaults do not match your requirements (see *Broker's Locale String Defaults*), we recommend you assign suitable locale string defaults for your country and region, see the respective attribute in *Codepage-specific Attributes* for how to customize the broker's locale string defaults.

To customize mapping of locale strings (optional)

- If the built-in locale string mapping mechanism does not match your requirements, you can assign specific codepages to locale strings. See *Broker's Built-in Locale String Mapping* and `locale-string` for information on customizing the mapping of locale strings to codepages.

Building and Installing ICU Custom Converters

User-written ICU custom-converters can be used for *ACI-based Programming*, *RPC-based Components*, and *Reliable RPC*. This section covers the following topics:

Writing a User-written ICU Converter

ICU uses algorithmic conversion, non-algorithmic conversion and combinations of both. See *ICU Conversion*. Non-algorithmic converters defined by the UCM format are the easiest way to define user-written ICU converters. See *UCM Format*.

To write a (non-algorithmic) user-written ICU converter

- Define the ICU converter file in UCM format using a text editor to meet your requirements.

  Note: For further explanation of the UCM file format, see *ICU Resources*.

Writing algorithmic and partially algorithmic converters can be complex. However, they can be installed into EntireX in the same way as the table-driven, non-algorithmic ones. A description of how to write algorithmic and partially algorithmic converters is beyond the scope of this documentation; please see the ICU documentation and other sources specified under *ICU Resources*.

Compiling a User-written ICU Converter

To compile the user-written ICU converter

- Compile the converter source files (extension *.ucm*) into binary converter files (extension ".cnv") using the ICU tool `makeconv`. Example:

  ```
  makeconv -v myebcdic.ucm
  ```

  Note: EntireX delivers the ICU tool `makeconv` in the EntireX *bin* directory.

  This produces a binary converter file named *myebcdic.cnv*.

  Caution: The binary format "cnv" depends on the endianness (big/little-endian) and character set family (ASCII/EBCDIC) of the computer where it is produced. For example, a binary converter file produced on a machine with big endianes cannot be executed

on a machine with little-endian (and vice versa) or character set family *EBCDIC* cannot be executed on a machine with character set family *ASCII* (and vice versa). It is highly recommended to compile the converter source file(s) on the same target platform where the broker runs - otherwise unpredictable result may occur.

Installing a User-written ICU Converter

To install the user-written ICU converter

1 Define the broker attribute `ICU-DATA-DIRECTORY`. See *Broker-specific Attributes*.

Example:

```
ICU DATA DIRECTORY="../EntireX/config/etb"
```

2 Define the subdirectory *icudt<icu-version><endianness>* within the `ICU-DATA-DIRECTORY`

where <*icu-version*> is the ICU version used, for example 54, and
<*endianness*> is either “b” (big-endian) or “l” (little-endian)

Examples:

```
../EntireX/config/etb/icudt54l
../EntireX/config/etb/icudt54b
```

Notes:

1. The subdirectory and its naming are given by ICU standard. It is not invented by Software AG.
2. See the Release Notes to determine the ICU version used by the broker you are running and form the correct directory name - otherwise the user-written ICU converter will not be located.
3. Take care to use the correct endianness given by the machine the broker is running on, otherwise the user-written ICU converter will not be located.
4. There are also other approaches supported by ICU to locate converters. These approaches are (also) ICU version dependent. However, Software AG recommends the mechanism described above. See the ICU website for more information under *ICU Resources*.

3 Copy the user-written ICU converter binary file (extension "cnv") to the directory referenced by `ICU-DATA-DIRECTORY` and its subdirectory defined under steps 1 and 2 above. Examples:

```
.../EntireX/config/etb/icudt54l/myebcdic.cnv
.../EntireX/config/etb/icudt54l/myascii.cnv
```

4 If the converter name is not sent as the locale string by your application, customize the mapping of locale strings by assigning the user-written ICU converter (codepage) to locale strings in the Broker attribute file, see `locale-string` for how to customize the mapping of locale strings to codepages. Example:

```
DEFAULTS=CODEPAGE
/* Customer-written ICU converter */
CP1140=myebcdic
CP0819=myascii
```

5 For the Broker attribute, check whether ICU conversion is possible, that is, the attribute `ICU-CONVERSION` is not defined (default=`YES`) or set to `YES`.

6 For the Broker attribute, check whether use of ICU custom converters is possible, that is, the attribute `ICU-SET-DATA-DIRECTORY` is not defined (default=`YES`) or set to `YES`.

Writing Translation User Exits

This section covers the following topics:

Introduction

EntireX Broker provides an interface to enable user-written translation routines in the programming language C. It contains three parameters:

- The address of the TRAP control block (TRAP = Translation Routine / Area for Parameters).
- The address of a temporary work area. It is aligned to fullword / long integer boundary (divisible by 4). The work area can only be used for temporary needs and is cleared after return.
- A fullword (long integer) that contains the length of the work area.

Note: Names for user-written translation routines starting with "SAG" are reserved for Software AG usage and must not be used, e.g. "SAGTCHA" and "SAGTRPC".

Structure of the TRAP Control Block

The C structure TR_TRAP covers the layout of the control block.

```
typedef struct _TR_TRAP                                    /* I / O */
{
  unsigned long  tr_type;             /* TRAP type: TRAP_TYPE  inp  */
#define TR_TYPE  2                    /* TRAP type ETB 121          */
  long           tr_ilen;             /* Input buffer length   inp  */
  unsigned char *tr_ibuf;             /* Ptr to input buffer   inp  */
  long           tr_olen;             /* Output buffer length  inp  */
  unsigned char *tr_obuf;             /* Ptr to output buffer  inp  */
  long           tr_dlen;             /* Len of data returned: out  */
                                      /* Minimum of tr_ilen         */
                                      /*        and tr_olen         */
  unsigned long  tr_shost;            /* Senders host          inp  */
#define TR_LITTLE_ENDIAN  0           /* little endian              */
#define TR_BIG_ENDIAN     1           /* big endian                 */
  unsigned long  tr_scode;            /* Senders character set inp  */
#define SEBCIBM ((1L << 5)|(1L << 1))  /* 0x22 EBCDIC (IBM)         */
#define SEBCSNI ((1L << 6)|(1L << 1))  /* 0x42 EBCDIC (SNI)         */
#define SA88591            (1L << 7)   /* 0x80 ASCII                */
  unsigned long  tr_rhost; /* Receivers host     (see tr_shost) inp  */
  unsigned long  tr_rcode; /* Receivers char set (see tr_scode) inp  */
  unsigned long  tr_bhost; /* BROKER host        (see tr_shost) inp  */
  unsigned long  tr_bcode; /* BROKER char set    (see tr_scode) inp  */
  unsigned long  tr_senva; /* Senders ENVIRONMENT field set:    inp  */
#define OFF  0                          /*  ENVIRONMENT field not set */
#define ON   1                          /*  ENVIRONMENT field set     */
  unsigned long  tr_renva; /* Receivers ENVIRONMENT field set:  inp  */
                           /* see tr_senva                          */
#define S_ENV 32                   /* size of ENVIRONMENT field     */
  char           tr_senv[S_ENV];/* Senders   ENVIRONMENT field  inp  */
  char           tr_renv[S_ENV];/* Receivers ENVIRONMENT field  inp  */
} TR_TRAP;
```

Using the TRAP Fields

The tr_dlen must be supplied by the user-written translation routine. It tells the Broker the length of the message of the translation. In our example its value is set to the minimum length of the input and output buffer.

All other TRAP fields are supplied by the Broker and must not be modified by the user-written translation routine.

The incoming message is located in a buffer pointed to by tr_ibuf. The length (not to be exceeded) is supplied in tr_ilen. The character set information from the send buffer can be taken from tr_scode.

The outgoing message must be written to the buffer pointed to by tr_obuf. The length of the output buffer is given in the field tr_olen. The character set is specified in tr_rcode. If the addresses given

in tr_ibuf and tr_obuf point to the same location, it is not necessary to copy the data from the input buffer to the output buffer.

The environment fields tr_senva and tr_renva are provided to handle site-dependent character set information. For the `SEND` and/or `RECEIVE` functions, you can specify data in the `ENVIRONMENT` field of the Broker ACI control block. This data is translated into the codepage of the platform where EntireX Broker is running (see field tr_bcode) and is available to the tr_senv or tr_renv field in the TRAP control block. tr_senva or tr_renva are set to `ON` if environmental data is available. Any values given in the API field `ENVIRONMENT` must correspond to the values handled in the translation routine.

Configuring Translation User Exits

To configure translation user exits

As a prerequisite, the user-written translation routine shared library/object must be accessible to the Broker worker threads.

1 Copy the user-written translation routine shared library/object into the EntireX *lib* directory.

2 In the Broker attribute file, set the service-specific attribute `TRANSLATION` to the name of the user-written translation routine. Example:

```
TRANSLATION=libmytrans.s[o|l]
```

or

1. Place the user-written translation routine shared library/object in a directory of your choice. Spaces in the path name are not allowed.
2. In the Broker attribute file, set the service-specific attribute `TRANSLATION` to the full path name of the directory of the user-written translation routine. Example:

```
TRANSLATION=../mydir/mytrans/libmytrans.s[o|l]
```

Writing SAGTRPC User Exits

This section covers the following topics:

Introduction

EntireX Broker provides an interface to SAGTRPC user exit routines written in the programming language C. The interface contains three parameters:

- The address of the UE (user exit) control block.
- The address of a temporary work area. It is aligned to a fullword / long-integer boundary (divisible by 4). The work area can only be used temporarily and is cleared after return.
- A fullword (long integer) that contains the length of the work area.

Note: Names for conversion routines starting with "SAG" are reserved for Software AG usage and must not be used, e.g. "SAGTCHA" and "SAGTRPC".

Structure of the User Exit Control Block

The C structure UECB shows the layout of the user exit control block.

```
typedef struct _UECB
{
   unsigned long          eVersion;
#define USRTRPC_VERSION_1               1

   char                *  pInputBuffer;
   unsigned long          uInputLen;
   char                *  pOutputBuffer;
   unsigned long          uOutputLen;
   unsigned long          uReturnedLen;

   unsigned long          shost;
#define USRTRPC_LITTLE_ENDIAN  0      /* little endian                    */
#define USRTRPC_BIG_ENDIAN     1      /* big endian                       */

   unsigned long          scode;
#define USRTRPC_SEBCIBM ((1L << 5)|(1L << 1)) /* 0x22 EBCDIC (IBM)        */
#define USRTRPC_SEBCSNI ((1L << 6)|(1L << 1)) /* 0x42 EBCDIC (SNI)        */
#define USRTRPC_SA88591            (1L << 7)  /* 0x80 ASCII               */

   unsigned long          rhost;
/* see shost */
   unsigned long          rcode;
/* see scode */
   unsigned long          bhost;
/* see shost */
   unsigned long          bcode;
/* see scode */

   unsigned long          uCpSender;
   unsigned long          uCpReceiver;
   unsigned long          uCpBroker;
```

```
   char                   eFunction;
#define USRTRPC_FCT_CONVERT             'C'
#define USRTRPC_FCT_GETLENGTH           'L'

   char                   eDirection;
#define USRTRPC_DIR_SENDER_TO_BROKER    '1'
#define USRTRPC_DIR_SENDER_TO_RECEIVER  '2'
#define USRTRPC_DIR_BROKER_TO_RECEIVER  '3'

   char                   sFormat[2];
#define ERX_USERDATA    "01"    /* UserId, Lib, Pgm, etc. from Header
                                   (truncatable)                          */
#define ERX_METADATA    "02"    /* Header Data (non-truncatable)          */
#define ERX_FRMTDATA    "03"    /* Format Buffer (non-truncatable)        */
#define ERX_SB_ELEMENT  "04"    /* String Buffer                          */
#define ERX_VB_METADATA "05"    /* Value Buffer Array Occurrences,
                                   String Length                          */
#define ERX_PREVIEW     "99"    /* Previewing FB and VB, etc...           */
                                /* Convert data lazy. Do not care on      */
                                /* length changes and truncation.         */
#define ERX_FRMT_A      "A "    /* Data Type A                            */
#define ERX_FRMT_AV     "AV"    /* Data Type AV                           */
#define ERX_FRMT_B      "B "    /* Data Type B                            */
#define ERX_FRMT_BV     "BV"    /* Data Type BV                           */
#define ERX_FRMT_D      "D "    /* Data Type D                            */
#define ERX_FRMT_F4     "F4"    /* Data Type F4                           */
#define ERX_FRMT_F8     "F8"    /* Data Type F8                           */
#define ERX_FRMT_I1     "I1"    /* Data Type I1                           */
#define ERX_FRMT_I2     "I2"    /* Data Type I2                           */
#define ERX_FRMT_I4     "I4"    /* Data Type I4                           */
#define ERX_FRMT_K      "K "    /* Data Type K                            */
#define ERX_FRMT_KV     "KV"    /* Data Type KV                           */
#define ERX_FRMT_L      "L "    /* Data Type L                            */
#define ERX_FRMT_N      "N "    /* Data Type N                            */
#define ERX_FRMT_P      "P "    /* Data Type P                            */
#define ERX_FRMT_T      "T "    /* Data Type T                            */
#define ERX_FRMT_U      "U "    /* Data Type U                            */
#define ERX_FRMT_UV     "UV"    /* Data Type UV                           */

   char                    szErrorText[40];

}  UECB;
```

The file *usrtrpc.c* is an example of the SAGTRPC user exit. It is delivered in the Broker user exit directory. See *Directories as Used in EntireX*.

Using the User Exit Interface Fields

The user exit provides two separate functions, `Convert` and `GetLength`. The field `eFunction` indicates the function to execute.

Errors

Both functions can send an error, using register 15 in the range 1 to 9999 to SAGTRPC together with an error text in the field `szErrorText`.

- A value of 0 returned in register 15 means successful response.
- Error 9999 is reserved for output buffer overflow. See *`Convert` Function*.
- When an error occurs, the conversion of the message will be aborted and the error text will be sent to the receiver (client or server). The error is prefixed with the error class 1011. See *Message Class 1011 - User-definable SAGTRPC Conversion Exit*.

Example:

The user exit returns 1 in register 15 and the message “Invalid Function” in `szErrorText`. The receiver gets the error message `10110001 Invalid Function`.

Convert Function

This function has to be executed when the contents of `eFunction` match the definition `USRTRPC_FCT_CONVERT`.

`uReturnedLen` must be supplied by SAGTRPC's user-written conversion exit. Its value must be set to the length of the output buffer.

All other interface fields are supplied by the Broker and must not be modified by SAGTRPC's user-written conversion exit.

The incoming data is located in a buffer pointed to by `pInputBuffer`. `uInputLen` defines the length.

The outgoing converted message must be written to the buffer pointed to by `pOutputBuffer`. The field tr_olen defines the maximum length available.

For variable length data such as AV and KV, an output buffer overflow can occur if the message size increases after conversion or the receiver's receive buffer is too small. In this case error 9999 “output buffer overflow” must be returned, which calls the *`GetLength` Function* for the remaining fields.

GetLength Function

The `GetLength` function evaluates the needed length of the output buffer after conversion. An actual conversion must not be performed. The length needed must be returned in the field `uOutputLen`.

The `GetLength` function is called for remaining fields after the `Convert` function returned the error 9999 “output buffer overflow”.

The purpose of this function is to evaluate the length needed by the receiver's receive buffer. This length is returned to the receiver in the ACI field `RETURN-LENGTH`. The receiver can then use the Broker ACI function `RECEIVE` with the option `LAST` together with a receive buffer large enough to reread the message.

Character Set and Codepage

The character-set information used is the same as in the user-written translation routine and is taken from `scode` (for the sender), `rcode` (for the receiver) and `bcode` (for the Broker). The character-set information depends on the direction information given in the field `eDirection`. See the following table:

eDirection	From Character Set	To Character Set
USRTRPC_DIR_SENDER_TO_BROKER	scode	bcode
USRTRPC_DIR_SENDER_TO_RECEIVER	scode	rcode
USRTRPC_DIR_BROKER_TO_RECEIVER	bcode	rcode

Alternatively, the codepage as derived from the locale string mapping process is provided in `uCpSender` (sender codepage), `uCpReceiver` (receiver codepage) and `uCpBroker` (Broker codepage), and can be used to find the correct conversion table. See the following table and also *Locale String Mapping*.

eDirection	From Codepage	To Codepage
USRTRPC_DIR_SENDER_TO_BROKER	uCpSender	uCpBroker
USRTRPC_DIR_SENDER_TO_RECEIVER	uCpSender	uCpReceiver
USRTRPC_DIR_BROKER_TO_RECEIVER	uCpBroker	uCpReceiver

Software AG IDL Data Types to Convert

The field `sFormat` provides the SAGTRPC user-written conversion exit with the information on the IDL data types to convert. Each data type can be handled independently.

sFormat	Data to be converted	Notes
FMTA	IDL data type A	1, 3, 4
FMTAV	IDL data type AV	4, 5
FMTB	IDL data type B	1, 2, 7
FMTBV	IDL data type BV	1, 2, 7
FMTD	IDL data type D	1, 2, 7
FMTF4	IDL data type F4	1, 2, 7
FMTF8	IDL data type F8	1, 2, 7
FMTI1	IDL data type I1	1, 2, 7
FMTI2	IDL data type I2	1, 2, 7
FMTI4	IDL data type I4	1, 2, 7
FMTK	IDL data type K	1, 3, 4
FMTKV	IDL data type KV	4, 5
FMTL	IDL data type L	1, 2, 7
FMTN	IDL data type N	1, 2, 7
FMTP	IDL data type P	1, 2, 7
FMTT	IDL data type T	1, 2, 8
FMTU	IDL data type U	1, 2, 7
FMTUV	IDL data type UV	1, 2, 7
FMTUSER	RPC user data such as user ID, library, program...	1, 3, 4
FMTMETA	RPC metadata	1, 2, 7
FMTFB	RPC format buffer	1, 2, 7
FMTSB	RPC metadata variable length	4, 5, 7
FMTPRE	Preview data	4, 6, 7

Notes:

1. Field length is constant.
2. The field content length must not increase or decrease during conversion. If this happens, the user exit should produce an error.
3. If the field content length *decreases* during the conversion, suitable padding characters (normally blanks) have to be used.
   If the field content length *increases* during conversion and exceeds the field length, the contents must be truncated or, alternatively, the conversion can be aborted and an error produced.

4. If the contents are truncated, character boundaries are the responsibility of the user exit. Complete valid characters after conversion have to be guaranteed. This may be a complex task for codepages described under *Arabic Shaping*, *EBCDIC Stateful Codepages* or *Multibyte or Double-byte Codepages*. For single-byte codepages it is simple because the character boundaries are the same as the byte boundaries.
5. The field length can decrease or increase during the conversion up to the output buffer length. The new field length must be returned in `uReturnedLen`. If the output buffer in the `Convert` function is too small, error 9999 must be returned to the caller.
6. The field buffer should continue to be converted until the output buffer is full or the input buffer has been processed. If the field content length increases or truncations occur, no error should be produced. If the field content length decreases, there should be no padding. The new field length should simply be returned to the caller.
7. Codepages used for RPC data streams must meet several requirements. See *Codepage Requirements for RPC Data Stream Conversions*. If these are not met, the codepage cannot be used to convert RPC data streams.

To compile and link the SAGTRPC user exit

- See the *README.TXT* in the *Broker User Exit Directory*.

Configuring SAGTRPC User Exits

The user-written SAGTRPC user exit shared library/object must be accessible to the Broker worker threads.

To configure SAGTRPC user exits

1 Copy the user-written SAGTRPC user exit shared library/object into the EntireX *lib* directory.

2 In the Broker attribute file, set the service-specific attribute `CONVERSION` to the name of your SAGTRPC user exit. Example:

```
CONVERSION=(libmytrans.s[o|l])
```

or

1. Place the user-written translation routine shared library/object in a directory of your choice.
2. In the Broker attribute file, set the service-specific attribute `CONVERSION` to the full path name of the directory of the SAGTRPC user exit. Example:

```
CONVERSION=../mydir/mytrans/libmytrans.s[o|l]
```

To configure locale string defaults

- If the broker's locale string defaults do not match your requirements, we recommend you assign suitable locale string defaults for your country and region. See the appropriate attribute under *Codepage-specific Attributes* for information on customizing broker's locale string defaults, and also *Locale String Mapping*.

To customize mapping of locale strings

- If the broker's built-in locale string mechanism does not match your requirements, you can assign specific codepages to locale strings. See *Broker's Built-in Locale String Mapping* and the appropriate attribute under *Codepage-specific Attributes* for information on customizing broker's locale string defaults.

6 Managing the Broker Persistent Store

The persistent store is used for storing unit-of-work messages to disk. This means message and status information can be recovered after a hardware or software failure to the previous commit point issued by each application component.

Under UNIX, the broker persistent store can be implemented with:

- the Adabas database of Software AG
- the c-tree (C) Copyright database of FairCom Corporation (R)

Note: If you were previously using the local file system of the machine where the Broker kernel executes, you will need to migrate to a supported persistent store. This persistent store option is no longer supported. To migrate your persistent store, please see the steps outlined in ***Migrating the Persistent Store***.

See also *Concepts of Persistent Messaging*.

Implementing an Adabas Database as Persistent Store

Introduction

EntireX provides an Adabas persistent driver. This enables Broker unit of work (UOW) messages and their status to be stored in an Adabas file. It is designed to work with Adabas databases under z/OS, UNIX, Windows, BS2000 and z/VSE, and can be used where the database resides on a different machine to Broker kernel. For performance reasons, we recommend using EntireX Broker on the same machine as the Adabas database.

Adabas Persistent Store Parameters

Parameters are supplied using the *Adabas-specific Attributes* in the platform-independent administration documentation. See excerpt from the attribute file:

```
DEFAULTS=BROKER
  STORE                     = BROKER
  PSTORE-TYPE               = ADABAS
  PSTORE                    = COLD

DEFAULTS=ADABAS
  DBID                      = dbid
  FNR                       = fnr
```

Configuring and Operating the Adabas Persistent Store

Selecting the Adabas Persistent Store Driver

The Adabas Persistent Store driver module is contained within the regular Broker load library or binaries directory. The module adapsi is activated by specifying the `PSTORE-TYPE` parameter as shown above.

Use the supplied script *persistence.fdu* in the *bin* directory to create a persistent store file in your Adabas database. This script uses the Adabas FDT definition found in file *persistence.fdt* in the *etc* directory.

The script *persistence.fdu* can be executed like this:

```
persistence.fdu <dbid> <fnr>
```

Note: You can customize the supplied script and FDT file in accordance with your site requirements. See the *Adabas Utilities* manual where necessary, specifically *ADAFDU (File Definition Utility)*.

To run the script file

1. Ensure that you execute the script file on the same machine that the target Adabas is running on. (The database can be either active or inactive at the time you execute it.)
2. Ensure that Adabas environment variables (such as `ACLDIR`, `ADATOOLS`, `ADABIN` and `ADALNK`) are set up. These environment variables are set by sourcing the corresponding environment scripts. See your Adabas documentation for details.
3. Set your working directory to the one where the fdt file is located.
4. Execute the fdt file, passing it two parameters. (The first one is the DBID, where persistent store file is to be created; the second is the file number.)
5. Option: If the DBID is less than 3 characters long, include leading zeros. For example:

   ```
   persistence.fdu 001 19
   ```

   Result: Creation of file number 19 in database 1.

Defining an Adabas FDT for EntireX File

```
ADACMP FNDEF='01,WK,21,A,DE'
ADACMP FNDEF='01,WJ,126,B,MU'
ADACMP FNDEF='01,WI,126,B,DE,NU'
ADACMP FNDEF='01,WL,96,A,DE,NU'
ADACMP FNDEF='01,WP,96,A,DE,NU'
```

Restrictions

If a HOT start is performed, the Broker kernel must be executed on the same platform on which also the previous Broker executed. This is because some portions of the persistent data are stored in the native character set and format of the Broker kernel. It is also necessary to start Broker with the same Broker ID as the previous Broker executed.

If a COLD start is executed, a check is made to ensure the Broker ID and platform information found in the persistent store file is consistent with the Broker being started (provided the persistent store file is not empty). This is done to prevent accidental deletion of data in the persistent store by a different Broker ID. If you intend to COLD start Broker and to utilize a persistent store file which has been used previously by a different Broker ID, you must supply the additional `PSTORE-TYPE` parameter `FORCE-COLD=Y`.

Recommendations

- Perform regular backup operations on your Adabas database. The persistent store driver writes C1 checkpoint records at each start up and shut down of Broker.
- For performance reasons, execute Broker on the same machine as Adabas.

Broker Checkpoints in Adabas

During startup, Broker writes the following C1 checkpoint records to the Adabas database. The time, date and job name are recorded in the Adabas checkpoint log. This enables Adabas protection logs to be coordinated with Broker executions. This information can be read from Adabas, using the `ADAREP` utility with option `CPLIST`:

Broker Execution Name	Broker Execution Type	Adabas
ETBC	Broker Cold Start	Normal Cold Start
ETBH	Broker Hot Start	Normal Hot Start
ETBT	Broker Termination	Normal Termination

Adabas DBA Considerations

BLKSIZE : Adabas Persistent Store Parameter for Broker

Caution should be exercised when defining the block size (`BLKSIZE`) parameter for the Adabas persistent store. This determines how much UOW message data can be stored within a single Adabas record. Therefore, do not define a much larger block size than the size of the maximum unit of work being processed by Broker. (Remember to add 41 bytes for each message in the unit of work.) The advantage of having a good fit between the unit of work and the block size is that fewer records are required for each I/O operation.

It is necessary to consider the following Adabas parameters and settings when using Adabas for the persistent store file:

Table of Adabas Parameter Settings

Topic	Description
Allowing Sufficient Adabas UQ Elements	Allow sufficient Adabas user queue (UQ) elements each time you start Broker. The Broker utilizes a number of user queue elements equal to the number of worker tasks (`NUM-WORKER`), plus two. Adabas timeout parameter (`TNAE`) determines how long the user queue elements will remain. This can be important if Broker is restarted after an abnormal termination, and provision must be made for sufficient user queue elements in the event of restarting Broker. Use either the Adabas utility `ADAOPR` or the Adabas DBA workbench to clean-up any user queue element belonging to the previous Broker job.
Setting Size of Hold Queue Parameters	Consideration must be given to the Adabas hold queue parameters `NISNHQ` and `NH`. These must be sufficiently large to allow Adabas to add/update/delete the actual number of records within a single unit of work. Example: where there are 100 message within a unit of work and the average message size is 10,000 bytes, the total unit of work size is 1 MB. If, for example, a 2 KB block size (default `BLKSIZE=2000`) is utilized by the Adabas persistent store driver, there will be 500 distinct records within a single Adabas commit (`ET`) operation, and provision must be made for this to occur successfully.

Topic	Description
Setting Adabas TT Parameter	Consideration must be given to the Adabas transaction time (`TT`) parameter for cases where a large number of records is being updated within a single unit of work.
Defining LWP Size	Sufficient logical work pool (`LWP`) size must be defined so that the Adabas persistent store can update and commit the units of work. Adabas must be able to accommodate this in addition to any other processing for which it is used.
Executing Broker Kernel and Adabas Nucleus on Separate Machines	If Broker kernel is executed on a separate machine to the Adabas nucleus, with a different architecture and codepage, then we recommend running the Adabas nucleus with the `UEC` (universal conversion) option in order to ensure that Adabas C1 checkpoints are legible within the Adabas checkpoint log.
Setting `INDEXCOMPRESSION=YES`	This Adabas option can be applied to the Adabas file to reduce by approximately 50% the amount of space consumed in the indexes.
4-byte ISNs	If you anticipate having more than 16 million records within the persistent store file, you must use 4-byte ISNs when defining the Adabas file for EntireX.
Specification of Adabas LP Parameter	**Caution:** This parameter must be specified large enough to allow the largest UOW to be stored in Adabas. If this is not large enough, Broker will detect an error (response 9; subresponse - 4 bytes - X'0003',C'LP') and Broker will not be able to write any further UOWs. See the description of the `LP` parameter under *ADARUN Parameters* in the*DBA Reference Summary* of the Adabas documentation.

Estimating the Number of Records to be Stored

To calculate the Adabas file size it is necessary to estimate the number of records being stored. As an approximate guide, there will be one Adabas record (500 bytes) for each unprocessed unit of work, plus also *n* records containing the actual message data, which depends on the logical block size and the size of the unit of work. In addition, there will be one single record (500 bytes) for each unit of work having a persisted status.

Always allow ample space for the Adabas persistent store file since the continuous operation of Broker relies of the availability of this file to store and retrieve information.

Note: If the Adabas file space is exceeded, Broker will automatically terminate, and it will be necessary either to increase the space available to the file using Adabas utilities or to perform a Broker HOT start with `NEW-UOW-MESSAGES=NO` to allow units of work to be consumed before normal operation can continue.

Estimating the Number of Records to be Stored

In this example there are 100,000 Active UOW records at any one time. Each of these is associated with two message records containing the message data. UOW records are 500 bytes in length. Each message record contains 2,000 bytes. In addition, there are 500,000 UOW status records residing in the persistent store, for which the UOW has already been completely processed. These are 500 bytes long.

Note: The actual size of the data stored within the UOW message records is the sum of all the messages within the UOW, plus a 41-byte header for each message. Therefore, if the average message length is 59 bytes, the two 2,000 bytes, messages records, could contain n = 4,000 / (59+41), or 40 messages. Adabas is assumed to compress the message data by 50% in the example (this can vary according to the nature of the message data).

3-byte ISNs and RABNs are assumed in this example. A device type of 8393 is used; therefore, the ASSO block size is 4,096, and DATA block size is 27,644. Padding factor of 10% is specified.

The following example calculates the space needed for Normal Index (NI), Upper Index (UI), Address Converter (AC) and Data Storage (DS).

Calculation Factors	Required Space
■ Number entries for descriptor WK (21-byte unique key)	■ = number UOW records: 0.1 + 0.5 million + number message records: 0.2 million
■ NI Space for descriptor WK ■ (3-byte ISN) ■ (4,092 ASSO block 10% padding)	■ = 800,000 * (3 + 21 + 2) ■ = 20,800,000 bytes ■ = 5,648 blocks
■ UI Space for descriptor WK ■ (3-byte ISN + 3-byte RABN) ■ (4,092 ASSO block 10% padding)	■ = 5,648 * (21 + 3 + 3 + 1) ■ = 158,140 bytes ■ = 43 blocks
■ Number entries for descriptor WI (8-byte unique key)	■ = number processed UOW records: 0.5 million
■ NI Space for descriptor WI ■ (3-byte ISN) ■ (4,092 ASSO block 10% padding)	■ = 500,000 * (3 + 8 + 2) ■ = 6,500,000 bytes ■ = 1,765 blocks
■ UI Space for descriptor WI ■ (3-byte ISN and 3 byte RABN) ■ (4,092 ASSO block 10% padding)	■ = 17,649 * (8 + 3 + 3 + 1) ■ = 26,475 bytes ■ = 8 blocks

Calculation Factors	Required Space
■ Number entries for descriptor WL (96 byte key)	■ = number UOW records 0.1 + 0.5 million
■ NI Space for descriptor WL ■ (3-byte ISN) ■ (4,092 ASSO block 10% padding)	■ = 600,000 * (3 + 96 + 2) ■ = 60,600,000 bytes ■ = 16,455 blocks
■ UI Space for descriptor WL ■ (3-byte ISN and 3 byte RABN) ■ (4,092 ASSO block 10% padding)	■ = 164,548 * (96 + 3 + 3 + 1) ■ = 16,948,517 bytes ■ = 461 blocks
■ Address Converter space ■ (4,092 ASSO block)	■ = (800,000 + 1) * 3 / 4092 ■ = 587 blocks
■ Data storage for message data (2,000-byte records compressed by 50%)	■ = 0.2 million * 2000 * 0.5 = 200,000,000 bytes
■ Data storage for UOW data (2,000-byte records compressed by 50%)	■ = 0.6 million * 500 * 0.5 = 150,000,000 byte
■ Combined space required for data (27,644 DATA block 10% padding)	■ = 14,068 blocks
Entity Requiring Space	**Total Required Space**
Normal Index (NI)	= 23,868 blocks
Upper Index (UI)	= 512 blocks
Data Storage (DS)	= 14,068 blocks
Address Converter (AC)	= 587 blocks

Tips on Transports, Platforms and Versions

- **Entire Net-Work**
  If you intend to use Adabas persistent store through Entire Net-Work, see the Entire Net-Work documentation for installation and configuration details.
- **Adabas Versions**
  Adabas persistent store can be used on all Adabas versions currently released and supported by Software AG.

- **Prerequisite Versions of Entire Net-Work with Adabas**
  See the Adabas and Entire Net-Work documentation to determine prerequisite versions of Entire Net-Work to use with Adabas at your site.

Copying the Persistent Store from/to another Adabas File or Database

The DBA can perform an `UNLOAD` of the Adabas file in which the persistent store is located (this must be done when Broker is not running). This allows the persistent store to be LOADed into another Adabas file, in the same or in another Adabas database. Broker can then be restarted (PSTORE=HOT) with the attributes specifying the new location of the persistent store file. See *Table of Adabas Parameter Settings* above. See separate Adabas documentation for details of Adabas utilities for `UNLOAD` and `LOAD` operations.

The persistent store file can only be reloaded into another Adabas database running on the same platform type as the Adabas database from which it was unloaded.

c-tree Database as Persistent Store

EntireX provides a c-tree© persistent driver based on the c-tree© User API of the FairCom Corporation®. This driver manages a fast and reliable embedded local database.

In order to operate EntireX using the c-tree persistent store option, you must assign Broker attribute `PSTORE-TYPE=CTREE`. No other attributes are required. However, several attributes are supported to set additional optional attributes for the c-tree store. See *c-tree-specific Attributes* for details.

Migrating the Persistent Store

The contents of EntireX Broker's persistent store can be migrated to a new persistent store in order to change the PSTORE type or to use the same type of PSTORE with increased capacity.

The migration procedure outlined here requires two Broker instances started with a special `RUN-MODE` parameter. One Broker unloads the contents of the persistent store and transmits the data to the other Broker, which loads data into the new PSTORE. Therefore, for the purposes of this discussion, we will refer to an *unload* Broker and a *load* Broker.

This procedure is based on Broker-to-Broker communication to establish a communication link between two Broker instances. It does not use any conversion facilities, since the migration procedure is supported for homogeneous platforms only.

Configuration

The migration procedure requires two Broker instances started with the `RUN-MODE` parameter. The unload Broker should be started with the following attribute:

```
RUN-MODE=PSTORE-UNLOAD
```

The load Broker should be started with the following attribute:

```
RUN-MODE=PSTORE-LOAD
```

These commands instruct the Broker instances to perform the PSTORE migration.

Note: The attribute `PARTNER-CLUSTER-ADDRESS` must be defined in both Broker instances to specify the transport address of the load Broker. The unload Broker must know the address of the load broker, and the load Broker must in turn know the address of the unload Broker.

Example:

Broker ETB001 performs the unload on host HOST1, and Broker ETB002 performs the load on host HOST2. The transmission is based on TCP/IP. Therefore, Broker ETB001 starts the TCP/IP communicator to establish port 1971, and Broker ETB002 starts the TCP/IP communicator to establish port 1972.

For ETB001, attribute `PARTNER-CLUSTER-ADDRESS=HOST2:1972:TCP` is set, and for ETB002, attribute `PARTNER-CLUSTER-ADDRESS=HOST1:1971:TCP` is set to establish the Broker-to-Broker communication between the two Broker instances.

In addition to attributes RUN-MODE and PARTNER-CLUSTER-ADDRESS, a fully functioning Broker configuration is required when starting the two Broker instances. To access an existing PSTORE on the unloader side, you must set the attribute PSTORE=HOT. To load the data into the new PSTORE on the loader side, you must set the attribute PSTORE=COLD. The load process requires an empty PSTORE at the beginning of the load process.

Note: Use caution not to assign PSTORE=COLD to your unload Broker instance, as this startup process will erase all data currently in the PSTORE.

For the migration process, the unload Broker and the load Broker must be assigned different persistent stores.

A report can be generated to detail all of the contents of the existing persistent store. At the end of the migration process, a second report can be run on the resulting new persistent store. These two reports can be compared to ensure that all contents were migrated properly. To run these reports, set the attribute PSTORE-REPORT=YES. See PSTORE for detailed description, especially for the file assignment.

Migration Procedure

The migration procedure is made up of three steps.

Step 1

The unload Broker and the load Broker instances can be started independently of each other. Each instance will wait for the other to become available before starting the unload/load procedure.

The unload Broker instance sends a handshake request to the load Broker instance in order to perform an initial compatibility check. This validation is performed by Broker according to platform architecture type and Broker version number. The handshake ensures a correctly configured partner cluster address and ensures that the user did not assign the same PSTORE to both Broker instances. If a problem is detected, an error message will be issued and both Broker instances will stop.

Step 2

The unload Broker instance reads all PSTORE data in a special non-destructive raw mode and transmits the data to the load Broker instance. The load Broker instance writes the unchanged raw data to the new PSTORE. A report is created if PSTORE-REPORT=YES is specified, and a valid output file for the report is specified.

Step 3

The unload Broker instance requests a summary report from the load Broker instance to compare the amount of migrated data. The result of this check is reported by the unload Broker instance and the load Broker instance before they shut down.

When a Broker instances is started in `RUN-MODE=PSTORE-LOAD` or `RUN-MODE=PSTORE-UNLOAD`, the Broker instances only allow Administration requests. All other user requests are prohibited.

Notes:

1. The contents of the persistent store are copied to the new persistent store as an exact replica. No filtering of unnecessary information will be performed, for example, UOWs in received state. The master records will not be updated.
2. Before restarting your Broker with the new persistent store, be sure to change your PSTORE attribute to `PSTORE=HOT`. *Do not* start your broker with the new persistence store using `PSTORE=COLD`; this startup process will erase all of the data in your persistent store.
3. After completing the migration process and restarting your broker in a normal run-mode, it is important not to bring both the new PSTORE and the old PSTORE back online using separate Broker instances; otherwise, applications would receive the same data twice. Once the migration process is completed satisfactorily, and is validated, the old PSTORE contents should be discarded.

7 Broker Resource Allocation

The EntireX Broker is a multithreaded application and communicates among multiple tasks in memory pools. If you do not need to restrict the memory expansion of EntireX Broker, we strongly recommend you enable the dynamic memory management in order to handle changing workload appropriately. See *Dynamic Memory Management* below. If dynamic memory management is disabled, non-expandable memory is allocated during startup to store all internal control blocks and the contents of messages.

General Considerations

Resource considerations apply to both the global and service-specific levels:

- Dynamic assignment of global resources to services that need them prevents the return of a "Resource Shortage" code to an application when resources are available globally. It also enables the EntireX Broker to run with fewer total resources, although it does not guarantee the availability of a specific set of resources for a particular service.
- Flow control ensures that individual services do not influence the behavior of other services by accident, error, or simply overload. This means that you can restrict the resource consumption of particular services in order to shield the other services.

In order to satisfy both global and service-specific requirements, the EntireX Broker allows you to allocate resources for each individual service or define global resources which are then allocated dynamically to any service that needs them.

The resources in question are the number of conversations, number of servers, plus units of work and the message storage, separated in a long buffer of 4096 bytes and short buffer of 256 bytes. These resources are typically the bottleneck in a system, especially when you consider that non-conversational communication is treated as the special case of "conversations with a single message only" within the EntireX Broker.

Global resources are defined by the parameters in the Broker section of the attribute file. The number of conversations allocated to each service is defined in the service-specific section of the attribute file. Because the conversations are shared by all servers that provide the service, a larger number of conversations should be allocated to services that are provided by more than one server. The number of conversations required is also affected by the number of clients accessing the service in parallel.

Specifying Global Resources

You can specify a set of global resources with no restrictions on which service allocates the resources:

- Specify the global attributes with the desired values.
- Do not specify any additional restrictions. That is, do not provide values for the following Broker-specific attributes:

```
LONG-BUFFER-DEFAULT
SHORT-BUFFER-DEFAULT
CONV-DEFAULT
SERVER-DEFAULT
```

- Also, do not provide values for the following server-specific attributes:

```
LONG-BUFFER-LIMIT
SERVER-LIMIT
SHORT-BUFFER-LIMIT
CONV-LIMIT
```

Example

The following example defines global resources. If no additional definitions are specified, resources are allocated and assigned to any server that needs them.

```
NUM-CONVERSATION=1000
NUM-LONG-BUFFER=200
NUM-SHORT-BUFFER=2000
NUM-SERVER=100
```

Restricting the Resources of Particular Services

You can restrict resource allocation for particular services in advance:

- Use `CONV-LIMIT` to limit the resource consumption for a specific service.
- Use `CONV-DEFAULT` to provide a default limit for services for which `CONV-LIMIT` is not defined.

Example

In the following example, attributes are used to restrict resource allocation:

```
DEFAULTS=BROKER
NUM-CONVERSATION=1000
CONV-DEFAULT=200

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A, CONV-LIMIT=100
CLASS=B, SERVER=B, SERVICE=B, CONV-LIMIT=UNLIM
CLASS=C, SERVER=C, SERVICE=C
```

- Memory for a total of 1000 conversations is allocated (`NUM-CONVERSATION=1000`).
- Service A (`CLASS A,SERVER A,SERVICE A`) is limited to 100 conversation control blocks used simultaneously (`CONV-LIMIT=100`). The application that wants to start more conversations than specified by the limit policy will receive a "Resource shortage" return code. This return code should result in a retry of the desired operation a little later, when the resource situation may have changed.
- Service B (`CLASS B,SERVER B,SERVICE B`) is allowed to try to allocate as many resources as necessary, provided the resources are available and not occupied by other services. The number of conversations that may be used by this service is unlimited (`CONV-LIMIT=UNLIM`).
- Service C (`CLASS C,SERVER C,SERVICE C`) has no explicit value for the `CONV-LIMIT` attribute. The number of conversation control blocks that it is allowed to use is therefore limited to the default value which is defined by the `CONV-DEFAULT` Broker attribute.

The same scheme applies to the allocation of message buffers and servers:

- In the following example, long message buffers are allocated using the keywords `NUM-LONG-BUFFER`, `LONG-BUFFER-DEFAULT` and `LONG-BUFFER-LIMIT`:

```
DEFAULTS=BROKER
NUM-LONG-BUFFER=2000
LONG-BUFFER-DEFAULT=250

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A, LONG-BUFFER-LIMIT=100
CLASS=B, SERVER=B, SERVICE=B, LONG-BUFFER-LIMIT=UNLIM
CLASS=C, SERVER=C, SERVICE=C
```

- In the following example, short message buffers are allocated using the keywords `NUM-SHORT-BUFFER`, `SHORT-BUFFER-DEFAULT` and `SHORT-BUFFER-LIMIT`:

```
DEFAULTS=BROKER
NUM-SHORT-BUFFER=2000
SHORT-BUFFER-DEFAULT=250

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A, SHORT-BUFFER-LIMIT=100
CLASS=B, SERVER=B, SERVICE=B, SHORT-BUFFER-LIMIT=UNLIM
CLASS=C, SERVER=C, SERVICE=C
```

- In the following example, servers are allocated using the keywords NUM-SERVER, SERVER-DEFAULT and SERVER-LIMIT:

```
DEFAULTS=BROKER
NUM-SERVER=2000
SERVER-DEFAULT=250

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A, SERVER-LIMIT=100
CLASS=B, SERVER=B, SERVICE=B, SERVER-LIMIT=UNLIM
CLASS=C, SERVER=C, SERVICE=C
```

Specifying Attributes for Privileged Services

If privileged services (services with access to unlimited resources) exist, specify UNLIMITED for the attributes CONV-LIMIT, SERVER-LIMIT, LONG-BUFFER-LIMIT and SHORT-BUFFER-LIMIT in the service-specific section of the attribute file.

For example:

```
DEFAULTS=SERVICE
CONV-LIMIT=UNLIM
LONG-BUFFER-LIMIT=UNLIM
SHORT-BUFFER-LIMIT=UNLIM
SERVER-LIMIT=UNLIM
```

To ensure a resource reservoir for peak load of privileged services, define more resources than would normally be expected by specifying larger numbers for the Broker attributes that control global resources:

```
NUM-SERVER
NUM-CONVERSATION
CONV-DEFAULT
LONG-BUFFER-DEFAULT
SHORT-BUFFER-DEFAULT
SERVER-DEFAULT
```

Maximum Units of Work

The maximum number of units of work (UOWs) that can be active concurrently is specified in the Broker attribute file. The MAX-UOWS attribute can be specified for the Broker globally as well as for individual services. It cannot be calculated automatically. If a service is intended to process UOWs, a MAX-UOWS value must be specified.

If message processing only is to be done, specify MAX-UOWS=0 (zero). The Broker (or the service) will not accept units of work, i.e., it will process only messages that are not part of a UOW. Zero is used as the default value for MAX-UOWS in order to prevent the sending of UOWs to services that are not intended to process them.

Calculating Resources Automatically

To ensure that each service runs without impacting other services, allow the EntireX Broker to calculate resource requirements automatically:

- Ensure that the attributes that define the default total for the Broker and the limit for each service are not set to UNLIM.
- Specify AUTO for the Broker attribute that defines the total number of the resource.
- Specify a suitable value for the Broker attribute that defines the default number of the resource.

The total number required will be calculated from the number defined for each service. The resources that can be calculated this way are Number of Conversations, Number of Servers, Long Message Buffers and Short Message Buffers.

Avoid altering the service-specific definitions at runtime. Doing so could corrupt the conversation consistency. Applications might receive a message such as "NUM-CONVERSATIONS reached" although the addressed service does not serve as many conversations as defined. The same applies to the attributes that define the long and short buffer resources.

Automatic resource calculation has the additional advantage of limiting the amount of memory used to run the EntireX Broker. Over time, you should be able to determine which services need more resources by noting the occurrence of the return code "resource shortage, please retry". You can then increase the resources for these services. To avoid disruption to the user, you could instead allocate a relatively large set of resources initially and then decrease the values using information gained from the Administration Monitor application.

Number of Conversations

To calculate the total number of conversations automatically, ensure that the CONV-DEFAULT Broker attribute and the CONV-LIMIT service-specific attribute are not set to UNLIM anywhere in the attribute

file. Specify NUM-CONVERSATION=AUTO and an appropriate value for the CONV-DEFAULT Broker attribute. The total number of conversations will be calculated using the value specified for each service.

For example:

```
DEFAULTS=BROKER
NUM-CONVERSATION=AUTO
CONV-DEFAULT=200

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A
CLASS=B, SERVER=B, SERVICE=B, CONV-LIMIT=100
CLASS=C, SERVER=C, SERVICE=C
```

- Service A and Service C both need 200 conversations (the default value). Service B needs 100 conversations (CONV-LIMIT=100).
- Because NUM-CONVERSATIONS is defined as AUTO, the broker calculates a total of 500 conversations (200 + 200 + 100).
- NUM-CONVERSATIONS=AUTO allows the number of conversations to be flexible without requiring additional specifications. It also ensures that the broker is started with enough resources to meet all the demands of the individual services.
- AUTO and UNLIM are mutually exclusive. If CONV-DEFAULT or a single CONV-LIMIT is defined as UNLIM, the EntireX Broker cannot determine the number of conversations to use in the calculation, and the EntireX Broker cannot be started.

Number of Servers

To calculate the number of servers automatically, ensure that the SERVER-DEFAULT Broker attribute and the SERVER-LIMIT service-specific attribute are not set to UNLIM anywhere in the attribute file. Specify NUM-SERVER=AUTO and an appropriate value for the SERVER-DEFAULT Broker attribute. The total number of server buffers will be calculated using the value specified for each service.

For example:

```
DEFAULTS=BROKER
NUM-SERVER=AUTO
SERVER-DEFAULT=250

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A, SERVER-LIMIT=100
CLASS=B, SERVER=B, SERVICE=B
CLASS=C, SERVER=C, SERVICE=C
```

Long Message Buffers

To calculate the number of long message buffers automatically, ensure that the LONG-BUFFER-DEFAULT Broker attribute and the LONG-BUFFER-LIMIT service-specific attribute are not set to UNLIM

anywhere in the attribute file. Specify NUM-LONG-BUFFER=AUTO and an appropriate value for the LONG-BUFFER-DEFAULT Broker attribute. The total number of long message buffers will be calculated using the value specified for each service.

For example:

```
DEFAULTS=BROKER
NUM-LONG-BUFFER=AUTO
LONG-BUFFER-DEFAULT=250

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A, LONG-BUFFER-LIMIT=100
CLASS=B, SERVER=B, SERVICE=B
CLASS=C, SERVER=C, SERVICE=C
```

Short Message Buffers

To calculate the number of short message buffers automatically, ensure that the SHORT-BUFFER-DEFAULT Broker attribute and the SHORT-BUFFER-LIMIT service-specific attribute are not set to UNLIM anywhere in the attribute file. Specify NUM-SHORT-BUFFER=AUTO and an appropriate value for the SHORT-BUFFER-DEFAULT Broker attribute. The total number of short message buffers will be calculated using the value specified for each service.

For example:

```
DEFAULTS=BROKER
NUM-SHORT-BUFFER=AUTO
SHORT-BUFFER-DEFAULT=250

DEFAULTS=SERVICE
CLASS=A, SERVER=A, SERVICE=A
CLASS=B, SERVER=B, SERVICE=B, SHORT-BUFFER-LIMIT=100
CLASS=C, SERVER=C, SERVICE=C
```

Dynamic Memory Management

Dynamic memory management is a feature to handle changing Broker workload without any restart of the Broker task. It increases the availability of the Broker by using various memory pools for various Broker resources and by being able to use a variable number of pools for the resources.

If more memory is needed than currently available, another memory pool is allocated for the specific type of resource. If a particular memory pool is no longer used, it will be deallocated.

The following Broker attributes can be omitted if DYNAMIC-MEMORY-MANAGEMENT=YES has been defined:

- NUM-CLIENT
- NUM-CMDLOG-FILTER
- NUM-COMBUF
- NUM-CONV[ERSATION]
- NUM-LONG[-BUFFER]
- NUM-SERVER
- NUM-SERVICE
- NUM-SERVICE-EXTENSION
- NUM-SHORT[-BUFFER]
- NUM-UOW|MAX-UOWS|MUOW
- NUM-WQE

If you want statistics on allocation and deallocation operations in Broker, you can configure Broker to create a storage report with the attribute STORAGE-REPORT. See *Storage Report* below.

Note: To ensure a stable environment, some pools of Broker are not deallocated automatically. The first pools of type COMMUNICATION, CONVERSATION, CONNECTION, HEAP, PARTICIPANT, PARTICIPANT EXTENSION, SERVICE ATTRIBUTES, SERVICE, SERVICE EXTENSION, TIMEOUT QUEUE, TRANSLATION, WORK QUEUE are excluded from the automatic deallocation even when they have not been used for quite some time. Large pools cannot be reallocated under some circumstances if the level of fragmentation in the address space has been increased in the meantime.

Dynamic Worker Management

Dynamic worker management is a feature to handle the fluctuating broker workload without re-starting the Broker task. It adjusts the number of running worker tasks according to current workload. The initial portion of worker tasks started at Broker startup is still determined by NUM-WORKER.

If more workers are needed than currently available, another worker task is started. If a worker task is no longer needed, it will be stopped.

The following Broker attributes are used for the configuration if DYNAMIC-WORKER-MANAGEMENT=YES has been defined:

- WORKER-MAX
- WORKER-MIN
- WORKER-NONACT
- WORKER-QUEUE-DEPTH
- WORKER-START-DELAY

The following two attributes are very performance-sensitive:

- Attribute WORKER-QUEUE-DEPTH defines the number of unassigned user requests in the input queue before a new worker task is started.

- Attribute WORKER-START-DELAY defines the time between the last worker task startup and the next check for another possible worker task startup. It is needed to consider the time for activating a worker task.

Both attributes depend on the environment, in particular the underlying operating system and the hardware. The goal is to achieve high-performance user request processing without starting too many worker tasks.

A good starting point to achieve high performance is not to change the attributes and to observe the performance of the application programs after activating the dynamic worker management.

If broker attribute DYNAMIC-WORKER-MANAGEMENT=YES is set, operator commands are available under z/OS to deactivate and subsequently reactivate dynamic worker management.

The following section illustrates the two different modes of dynamic worker management:

- **Scenario 1**

```
DYNAMIC-WORKER-MANAGEMENT=YES
NUM-WORKER = 5
WORKER-MIN = 1
WORKER-MAX = 32
```

  Broker is started with 5 worker tasks and then dynamically varies the number of worker tasks within the range from WORKER-MIN=1 to WORKER-MAX=32 due to DYNAMIC-WORKER-MANAGEMENT=YES.

- **Scenario 2**

```
DYNAMIC-WORKER-MANAGEMENT=NO
NUM-WORKER = 5
WORKER-MIN = 1
WORKER-MAX = 32
```

  Broker is started with 5 worker tasks. The WORKER-MIN/MAX attributes are ignored due to DYNAMIC-WORKER-MANAGEMENT=NO.

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.

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

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.

Sample Storage Report

The following is an excerpt from a sample STORAGE report.

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

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

Header	Description
Identifier	Name of the memory pool.
Address	Start address of the memory pool.
Size	Size of the memory pool.
Total	Total size of all obtained memory pools.
Date, Time	Date and time of the action.

Header	Description
Action	The action of Broker. The following actions are currently supported: Allocated: memory pool is allocated. Deallocated: memory pool is deallocated.

Maximum TCP/IP Connections per Communicator

This table shows the generated maximum number of TCP/IP connections per communicator. See also:

- Note for Linux

Platform	Maximum Number of TCP/IP Connections per Communicator
AIX	2,048
BS2000	2,048
Linux	65,534
Solaris	65,356
Windows	4,096
z/OS	16,384
z/VSE	2,048

With the Broker-specific attribute `POLL`, these restrictions can be lifted under z/OS, UNIX and z/VSE. See `POLL`.

The number of communicators multiplied by the maximum number of connections cannot exceed the maximum number of file descriptors per process.

See also `MAX-CONNECTIONS` under `TCP-OBJECT (Struct INFO_TCP)` under *Broker CIS Data Structures* in the EntireX Broker ACI Programming documentation.

Note for Linux

Under Linux, setting the maximum open file limit depends on your working environment:

- bash
- systemd

bash

In the bash shell you can display or change the limits with the command `ulimit -n`. These limits are used when the Broker (etbnuc) is started from the command line or from a cron job.

The limits can be stored, for example, in the file */etc/security/limits.conf*.

- For all users:

```
* soft nofile 1024
* hard nofile 8192
```

- For user entirex:

```
entirex soft nofile   8192
entirex hard nofile 100000
```

Broker uses the soft limit. When this limit is reached, no more connections are possible. If the hard limit is higher than the soft limit, you can increase the limit - without having to stop the broker - using the following command:

```
#> prlimit --pid <pid> --nofile = 4096:8192
```

The maximum limit in the broker for `POLL=NO` is 65534. `POLL=YES` is not subject to any limit and is dependent only on the soft limit of the system.

systemd

If the broker is controlled by a service that was started by `systemd`, the limits of `systemd` apply.

There are various ways of increasing the limits if you need more than 4096 connections:

- Set `DefaultLimitNOFILE` in the files */etc/systemd/system.conf* or */etc/systemd/user.conf*.
- Insert `LimitNOFILE=<new-limit>` in a service file */usr/lib/systemd/system/sag<n>exx<vers>*. Example:

```
# ----------------------------------------------------------------------------
# Copyright (c) 2014-2020 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.
# Use, reproduction, transfer, publication or disclosure is prohibited except
# as specifically provided for in your License Agreement with Software AG.
# ----------------------------------------------------------------------------
# do not modify this line
[Unit]
Description=sag7exx105
After=multi-user.target

[Service]
Type=forking
RemainAfterExit=yes
PrivateTmp=no
KillMode=none
TimeoutStartSec=330
TimeoutStopSec=330
LimitNOFILE=32000
User=rdsadmin
```

```
Group=rdstst
ExecStart=/bin/sh -c "/opt/testenv/exx/105/installed/EntireX/bin/sagexx105 start"
ExecStop=/bin/sh -c "/opt/testenv/exx/105/installed/EntireX/bin/sagexx105 stop"
PIDFile=/opt/testenv/exx/v105/installed/EntireX/bin/sagexx105.pid

[Install]
WantedBy=multi-user.target
```

You can check the current settings using the proc file system:

```
#> cat /proc/<etbnuc-pid>/limits

Limit                     Soft Limit           Hard Limit           Units
Max cpu time              unlimited            unlimited            seconds
Max file size             unlimited            unlimited            bytes
Max data size             unlimited            unlimited            bytes
Max stack size            8388608              unlimited            bytes
Max core file size        0                    unlimited            bytes
Max resident set          unlimited            unlimited            bytes
Max processes             15709                15709                processes
Max open files            32000                32000                files
Max locked memory         65536                65536                bytes
Max address space         unlimited            unlimited            bytes
Max file locks            unlimited            unlimited            locks
Max pending signals       15709                15709                signals
Max msgqueue size         819200               819200               bytes
Max nice priority         0                    0
Max realtime priority     0                    0
Max realtime timeout      unlimited            unlimited            us
```

8 Administering Broker Stubs

Available Stubs

The following table lists available stubs and gives an overview of available features and supported transport methods.

Stub	Language	Transport Methods	More Information
Jaci	Java	TCP /SSL	See *EntireX Java ACI*.
broker.s[o|l]	C	TCP / SSL	See below.

Transport Methods for Broker Stubs

The Broker stub can use TCP/IP and SSL. The term "SSL" in this section refers to both SSL (Secure Sockets Layer) and TLS (Transport Layer Security).

Using TCP/IP as Transport Method for the Broker Stub

To use TCP/IP

1 Optional: set the timeout, see ***Setting the Timeout for the Transport Method***.

2 The Broker stub requires the IP address and the TCP port number (if the Broker's default TCP port number 1971 cannot be used) for each BROKER-ID. Either add an entry in the Domain Name System (DNS) or modify your local hosts and services tables. See ***Modifying the Hosts and Services Tables***.

You can check whether the Broker has already been added to your DNS with the command:

```
ping <broker-id>
```

for example: ping ETB001. If a message such as "...is alive" or "Reply from ..." is displayed (the text displayed varies depending on your ping implementation), the name is known to your DNS and the host where the Broker is running is reachable. However, this does not necessarily mean that the Broker is active.

Using SSL/TLS as Transport Method for the Broker Stub

ACI applications 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. ACI-based clients or servers are always SSL clients. The SSL server can be either the EntireX Broker or the Broker SSL Agent. For an introduction see *SSL/TLS and Certificates with EntireX* in the Platform-independent Administration documentation.

With the Broker ACI, the SSL parameters (e.g. certificates) are provided with the function `SETSSLPARMS`.

To use SSL

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

2 Specify the Broker ID, using one of the following styles:

- *URL Style*, for example:

```
ssl://localhost:2010
```

- *Transport-method Style*, for example:

```
ETB024:1609:SSL
```

If no port number is specified, port 1958 is used as default.

3 Specify SSL parameters in the second parameter, for example:

```
'broker' etbcb "VERIFY_SERVER=N&TRUST_STORE=c:\\certs\\CaCert.pem"
```

If the SSL client checks the validity of the SSL server only, this is known as *one-way SSL*. The mandatory `trust_store` parameter specifies the file name of a keystore that must contain the list of trusted certificate authorities for the certificate of the SSL server. By default a check is made that the certificate of the SSL server is issued for the hostname specified in the Broker ID. The common name of the subject entry in the server's certificate is checked against the hostname. If they do not match, the connection will be refused. You can disable this check with SSL parameter `verify_server=no`.

If the SSL server additionally checks the identity of the SSL client, this is known as *two-way SSL*. In this case the SSL server requests a client certificate (the parameter `verify_client=yes` is defined in the configuration of the SSL server). Two additional SSL parameters must be specified on the SSL client side: `key_store` and `key_passwd`. This keystore must contain the

private key of the SSL client. The password that protects the private key is specified with `key_passwd`.

The ampersand (&) character cannot appear in the password.

SSL parameters are separated by ampersand (&). See also *SSL/TLS Parameters for SSL Clients*.

4 Make sure the SSL server to which the ACI side connects is prepared for SSL connections as well. The SSL server can be EntireX Broker or Broker SSL Agent. See:

- *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation
- Broker SSL Agent in the UNIX and Windows Administration documentation

Notes

- See table *Using SSL/TLS with EntireX Components* if SSL is required for other EntireX components.
- The Broker stub requires the IP address and the SSL port number for each `BROKER-ID`. Either add an entry to the Domain Name System (DNS) or modify your local hosts and services tables. See ***Modifying the Hosts and Services Tables***.

  If no port number is specified, port 1958 is used as default.
- You can check whether the Broker has already been added to your DNS with a `ping <broker-id>` command, for example:

```
ping ETB001
```

  If a message such as "`...is alive`" or "`Reply from ...`" is displayed (the text displayed varies depending on your ping implementation), the name is known to your DNS and the host where the Broker is running is reachable. However, this does not necessarily mean that the Broker is active.
- Take care if trace is switched on:

  Caution: If stub tracing level is > 1, unencrypted contents of the send/receive buffers are exposed in the trace.

- Example on running the delivered ACI example:

```
C:\SoftwareAG\EntireX\examples\ACI\conversational\C\convSvr -blocalhost:1958:SSL
-cACLASS -sASERVER -vASERVICE
-x"VERIFY_SERVER=N&TRUST_STORE=C:\SoftwareAG\EntireX\etc\ExxCACert.pem"
```

Setting the Timeout for the Transport Method

The timeout settings of the transport layers are independent of the broker's timeout settings, which are set by the application in the `WAIT` field of the broker ACI control block.

If the transport layer is interrupted, communication between the Broker and the stub (i.e. client or server application) is interrupted as well. To prevent a client from waiting for a Broker reply indefinitely, set a timeout value for the transport method. The actual timeout for the procedure is then the Broker timeout (which is set by the application in the `WAIT` of the broker ACI control block) plus this value.

To set a transport timeout value

- Set the environment variable `ETB_TIMEOUT`:

Transport Timeout Value	Description
0	Infinite wait for the application.
n	Transport method waits additional time in seconds. A negative value is treated as `ETB_TIMEOUT=0` (infinite wait).
No environment variable defined	Transport method waits additional 20 seconds.

See also *UNIX Commands to Set the Environment Variables*.

Limiting the TCP/IP Connection Lifetime

With transport method TCP/IP, the broker stub establishes one or more TCP/IP connections to the brokers specified with `BROKER-ID`. These connections can be controlled by the transport-specific `CONNECTION-NONACT` attribute on the broker side, but also by the transport-specific environment variable `ETB_NONACT` on the stub side. If `ETB_NONACT` is not 0, it defines the non-activity time (in seconds) of active TCP/IP connections to any broker. See `ETB_NONACT` under *Environment Variables in EntireX*. Whenever the broker stub is called, it checks for the elapsed non-activity time and closes connections with a non-activity time greater than the value defined with `ETB_NONACT`. Stubs capable of running in SRB mode do not support `ETB_NONACT` handling.

Transport Non-activity Value	Description
0	Infinite lifetime until application is stopped.
n (seconds)	Transport connections with non-activity time greater than *n* will be closed.
Nothing set	Infinite lifetime until application is stopped.

Modifying the Hosts and Services Tables

The Hosts and Services tables are plain text files in directory */etc*.

To add an entry to the hosts table

- Add a line similar to the following to the local hosts file:

```
100.100.1.1 ETB226 # ETB test host name
```

To add an entry to the services table

- Add lines similar to the following to the local services file:

```
ETB226 18492/tcp # ETB test host name
ETB411 21234/tcp # ETB production host name
```

Tracing for Broker Stubs

The broker stubs provide an option for writing trace files.

To switch on tracing for the broker stub

- Before starting the client application, set the environment variable `ETB_STUBLOG`:

Trace Value	Trace Level	Description
0	`NONE`	No tracing.
1	`STANDARD`	Traces initialization, errors, and all ACI request/reply strings.
2	`ADVANCED`	Used primarily by system engineers, traces everything from level 1 and provides additional information - for example the Broker ACI control block - as well as transport information.
3	`SUPPORT`	This is full tracing through the stub, including detailed traces of control blocks, message information, etc.

Example:

```
ETB_STUBLOG=2
```

If the trace level is greater than 1, unencrypted contents of the send/receive buffers may be exposed in the trace.

The trace file is created in the current directory. The name is *pid.etb* where *pid* is the process ID. If you want to write the trace file to a different location, set environment variable ETB_STUBLOGPATH to the desired path.

See also ***UNIX Commands to Set the Environment Variables***.

Remember to switch off tracing to prevent trace files from filling up your disk.

To switch off tracing for the broker stub

- Set the environment variable ETB_STUBLOG to NONE or delete it.

Application Stublog File

Logging works for both TCP and SSL. Tracing is controlled by the environment variable ETB_STUBLOG.

csh or tcsh users use:

```
setenv ETB_STUBLOG tracelevel
```

bsh, ksh or bash users use:

```
ETB_STUBLOG=tracelevel; export ETB_STUBLOG
```

Possible values for *tracelevel*:

Trace Value	Trace Level	Description
0	NONE	No tracing.
1	STANDARD	Traces initialization, errors, and all ACI request/reply strings.
2	ADVANCED	Used primarily by system engineers, traces everything from level 1 and provides additional information - for example the Broker ACI control block - as well as transport information.
3	SUPPORT	This is full tracing through the stub, including detailed traces of control blocks, message information, etc.

If you start your application with this environment variable set, a log file is created in the directory where your application is started. The name of the log file is *pid.etb*

csh or tcsh users use:

```
unsetenv ETB_STUBLOG
```

bsh, ksh or bash users use:

```
unset ETB_STUBLOG
```

UNIX Commands to Set the Environment Variables

Example of `ETB_TRANSPORT`:

Shell	set the environment variable:	delete the environment variable:
C Shell	`setenv ETB_TRANSPORT value`	`unsetenv ETB_TRANSPORT`
Bourne or Korn Shell	`ETB_TRANSPORT=value` `export ETB_TRANSPORT`	`unset ETB_TRANSPORT`

Support of Clustering in a High Availability Scenario

EntireX Broker supports clustering in a high-availability scenario, using the environment variable `ETB_SOCKETPOOL`. See *Environment Variables in EntireX*. This section covers the following topics:

- Introduction
- Exceptions
- Default

See also *High Availability in EntireX*.

Introduction

A TCP/IP connection established between stub and broker is not exclusively assigned to a particular thread. With multithreaded applications, two or more threads may use the same connection. On the other hand, if a connection is busy, another new one is created to exchange data.

In order to access the same broker instance in a clustering environment, an affinity between application thread and TCP/IP connection is needed to always use the same connection within an application thread. Therefore, an environment variable is evaluated to control the handling of TCP/IP connections.

If environment variable `ETB_SOCKETPOOL` is set to "OFF" (`ETB_SOCKETPOOL=OFF`), an affinity between threads and TCP/IP connections is established. All requests to one particular broker will use the same TCP/IP connection. `ETB_SOCKETPOOL` controls all TCP/IP connections.

Stubs ARFETB and NATETBZ always establish an affinity between subtask and TCP/IP connection.

Exceptions

Broker attribute `CONNECTION-NONACT` is used by the broker to close TCP/IP connections after the elapsed non-activity time. Omit this attribute to keep the TCP/IP connection alive.

Default

`ETB_SOCKETPOOL=ON` is the default setting. In this case, an established broker connection can be used by any thread if the connection is not busy.

9 Broker Command-line Utilities

EntireX Broker provides the following internal services: Command Service, Information Service and Administration Service, which can be used to administer and monitor brokers and RPC servers. Because these services are implemented internally, nothing has to be started or configured. You can use these services immediately after starting EntireX Broker.

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. `etbinfo` uses data descriptions called profiles to control the type of data that is returned for a request. `etbinfo` is useful for monitoring and administering EntireX Broker efficiently, for example how many users can run concurrently and whether the number of specified message containers is large enough.

Although basic formatting of the output is available, it is usually formatted by script languages or other means external to the Broker.

Running the Command-line Utility

In a UNIX environment, run the command-line utility with `etbinfo`. If the environment variable `LOGNAME` is not set, you must use the -x option (see below) to provide a user ID if the Broker is running with EntireX Security. `etbinfo` is located in directory */<Install_Dir>/EntireX/bin*.

Command-line Parameters

The table below explains the command-line parameters. The format string and profile parameters are described in detail following the table. All entries in the Option column are case-sensitive.

Option	Command-line Parameter	Req/ Opt	Explanation
`-b`	`brokerid`	R	Broker identifier, for example `localhost:1971:TCP`.
`-c`	`class`	O	Class as selection criterion.
`-C`		O	Create output with comma-separated values, suitable for input into a spreadsheet or other analysis tool. Any format string specified by means of format string or profile command-line parameters is ignored.

Option	Command-line Parameter	Req/ Opt	Explanation
-d	object	R	Possible values: Object — Provides Info on BROKER — Broker. CLIENT — Client. CMDLOG-FILTER — Command log filter. CONVERSATION — Conversation. NET — Entire Net-Work transport. PARTICIPANT — Participant. POOL-USAGE — Broker pool usage. PSF — Unit-of-work status. PSFADA — Adabas persistent store. PSFCTREE — c-tree persistent store. PSFDIV — DIV persistent store. RESOURCE-USAGE — Broker resource usage. SECURITY — EntireX Security. SERVER — Server. SERVICE — Service. SSL — SSL transport. STATISTICS — Broker statistics. TCP — TCP transport. UOW-STATISTICS — Units of work per service. USER — Participant (short). WORKER — Worker. WORKER-USAGE — Worker usage.
-e	recv class	O	Receiver's class name. This selection criterion is valid only for object PSF.
-f	*Format String*	O	Format string how you expect the output. See *Profile*.
-g	recv service	O	Receiver's service name. This selection criterion is valid only for object PSF.
-h	help	O	Prints help information.
-i	convid	O	Conversation ID as selection criterion. Only valid for object CONVERSATION.
-I	conv type	O	Conversation's type.
-j	recv server	O	Receiver's server name. This selection criterion is valid only for object PSF.

Option	Command-line Parameter	Req/ Opt	Explanation
-k	recv token	O	Receiver's token. This selection criterion is valid only for object PSF.
-l	level	O	The amount of information displayed: FULL All information. SHORT User-specific information.
-m	recv userid	O	Receiver's user ID. This selection criterion is valid only for object PSF.
-n	server name	O	Server name. This selection criterion is valid only for the objects SERVER, SERVICE or CONVERSATION.
-p	file	O	Here you can specify a file that defines the layout of the output. There are default files you can modify or you can use your own. The default files are: BROKER CLIENT CLOGFLT CONV NET POOL PSF PSFADA PSFCTREE PSFDIV SERVICE SSL STATIS STATIS TCP USER WORKER WKRUSAGE See *Profile*.
-q	puserid	O	Physical user ID. This selection criterion is valid only for objects CLIENT, SERVER, CONVERSATION, **Note:** Must be a hex value.
-r	sec	O	Refresh information after seconds.
-s	service	O	Service. This selection criterion is valid only for objects SERVER, SERVICE or CONVERSATION.
-S	"sslparms"	O	When using SSL transport for Broker communication. See *Using SSL/TLS*.
-t	token	O	This selection criterion is valid only for objects CLIENT, SERVER, SERVICE or CONVERSATION.
-u	userid	O	User ID. This selection criterion is only valid for the display types CLIENT, SERVER, SERVICE or CONVERSATION.
-v	UOW status	O	Unit of work status. This selection criterion is valid only for object PSF.
-w	UOW ID	O	Unit of work ID. This selection criterion is valid only for object PSF.
-x	userid	O	User ID. For security purposes.
-y	password	O	Password. For security purposes.
-z	token	O	Used with userid to uniquely identify a caller to Command and Information Services.

Option	Command-line Parameter	Req/ Opt	Explanation
`--longmsg`		O	If an error occurs, delivers the long text of an error message, corresponding to *Error Messages and Codes*. Output is generated as with the `exxmsg` utility. See *EXXMSG - Command-line Tool for Displaying Error Messages* in the Error Messages and Codes documentation.
`--external`		O	Reduces the output of `SERVICE` objects to external services. Broker-internal services are not displayed.
`--internal`		O	Reduces the output of `SERVICE` objects to Broker-internal services. The external user-specific services are not displayed.
`--pingrpc`		O	Executes an RPC ping to a specified RPC service. The parameters `-c <class_name>`, `-n <server_name>` and `-s <service>` are also required. If the service is running, return code 0 and a corresponding text are returned. If the service is not running, a return code other than 0 is given.
`--encrypted_password_from_stdin`		O	Encrypted password. See ***Using an Encrypted Password***.

Command-line Parameters from File

`etbinfo` supports an alternative method of passing command-line parameters.

If the environment variable `INF_ATTR` is set, the content is interpreted as a file name. If no command-line parameters are given, the command `etbinfo` evaluates the content of the file. Example:

```
-blocalhost:3930:TCP
-dBROKER
```

Profile

If you do not use the profile option or a format string, your output will be an unformatted list with all columns of that display type. To display specific columns, specify a profile that includes only those columns.

The following default sample profiles include all the columns defined for each display type:

- `BROKER`
- `CLIENT`
- `CLOGFLT`
- `CONV`
- `NET`
- `POOL`
- `PSF`
- `PSFADA`
- `PSFCTREE`
- `PSFDIV`
- `RESOURCE`
- `SECURITY`
- `SERVER`
- `SERVICE`
- `SSL`
- `STATIS`
- `TCP`
- `USER`
- `WKRUSAGE`
- `WORKER`

You can either delete the columns not required or copy the default profile and modify the order of the columns. Ensure that the column names have a leading "%". Column names can be written

in one line or on separate lines. The output is always written side by side. With profile parameters `%DATE` and `%TIME` you can provide a timestamp for the command-line query.

Location of Profiles

On UNIX, the profiles are contained in directory */<Install_Dir>/EntireX/etc* and are named *broker.pro, client.pro* etc.

Example 1

Profile for object SERVICE: SERVICE.

```
etbinfo -b ETB001 -d SERVICE -p service.pro -l FULL
```

The following list is displayed:

```
SAG                                ETBCIS     INFO
 1 0 16 86400 0 31647 0 00 00 00 00 0 0
SAG                                ETBCIS     USER-INFO
 2 0 16 86400 0 31647 0 00 00 00 00 0 0
SAG                                ETBCIS     CMD
 6 0 16 86400 0 31647 0 00 00 00 00 0 0
```

Example 2

Your own profile: MYPROF

```
etbinfo -b ETB001 -d SERVICE -p my_service.pro
```

Note: In this case, `my_service.pro` contains:`%4.4SERVERCLASS %SERVERNAME`

The following list is displayed:

```
ACLA    ASERVER
BCLA    BSERVER
CCLA    CSERVER
```

Sample Profiles for etbinfo

You can find the sample profiles for `etbinfo` in your */<Install_Dir>/EntireX/config* directory.

Format String

The format string, if specified, will override the use of a profile. The format string is built like a `printf()` in C language. The string must be enclosed in quotation marks. You can specify the columns by using a "%" and the column name. The column name must contain letters only. Numeric characters are not allowed. You can specify the length of column output by using a format precision, as in the ANSI-C `printf()` function. The column name must be followed by a blank. For example:

```
etbinfo -b ETB001 -d BROKER -f "%12.12CPLATNAME   %NUM-SERVER      %NUM-CLIENT"
```

which produces the following output, for example:

```
MVS/SP 7.04 30 100
```

You can also use an arbitrary column separator, which can be any character other than "%". You can use `\n` for a new line in the output and `\t` for a tabulator in the format string or profile. For example:

```
etbinfo -b ETB001 -d SERVER -f "UserID: %5.5USER-ID Token: %5.5TOKEN"
```

which produces:

```
UserID: HUGO  Token: MYTOK
UserID: EGON  Token:
UserID: HELMU Token: Helmu
```

If you want to structure your output a little more, you can operate with the `\n` or `\t` character. For example:

```
etbinfo -b ETB001 -d SERVICE -f  "Class:%5.5SERVER-CLASS \n\tName:%5.5SERVER-NAME ↩
\n\tService:%5.5SERVICE"
```

which produces:

```
Class:DATAB
    Name:DB10
    Service:Admin
Class:PRINT
    Name:LPT1
    Service:PRINT
...
```

You can also add a timestamp to the query:

```
etbinfo -b ETB001 -d BROKER -f "%DATE %TIME"
```

which produces:

```
2014-08-19 10:00:00.234
```

Using SSL/TLS

To set up SSL

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

2 Specify the Broker ID, using one of the following styles:

- *URL Style*, for example:

  ```
  ssl://localhost:2010
  ```

- *Transport-method Style*, for example:

  ```
  ETB024:1609:SSL
  ```

  If no port number is specified, port 1958 is used as default.

3 Specify SSL parameters with the option -s|S (lowercase for `etbcmd`; uppercase for `etbinfo`). See *SSL/TLS Parameters for SSL Clients*.

4 Make sure the broker is prepared for SSL connections as well. See *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation.

Using an Encrypted Password

You can encrypt a password and store this in a file. Specify this file instead of a cleartext password when you call a secure broker.

Note: We strongly recommend that your cleartext password is longer than 16 characters.

To encrypt a password

1 Enter the command:

```
etbnattr --echo_password_only -w clear_text_password ↩
```

The encrypted password is written to stdout.

2 Copy the password value to an empty file. (Ignore the prefix `KEY-PASSWD-ENCRYPTED:`.)

To specify the encrypted password from stdin

- Enter the command:

```
etbinfo -x uid --encrypted_password_from_stdin < file
```

Where *file* is the file containing the encrypted password you created as described above. Example:

```
etbinfo -b localhost:1971 -d BROKER -x UID --encrypted_password_from_stdin < myPwd
```

etbcmd

Allows the user to take actions - for example purge a unit of work, stop a server, shut down a Broker - against EntireX Broker.

Running the Command-line Utility

In a UNIX environment, run the command-line utility with `etbcmd`. If the environment variable `LOGNAME` is not set, you must use the `-x` option (see below) to provide a user ID if the Broker is running with EntireX Security. `etbcmd` is located in the directory */<Install_Dir>/EntireX/bin*.

Command-line Parameters

The table below explains the command-line parameters. All entries in the **Option** column are case-sensitive.

Command-line Parameter	Option	Parameter	Req/ Opt	Explanation
`brokerid`	`-b`	e.g. `ETB001`	R	Broker ID.
`command`	`-c`	■ `ALLOW-NEWUOWMSGS` ■ `APPMON-ON` ■ `APPMON-OFF` ■ `CLEAR-CMDLOG-FILTER` ■ `CONNECT-PSTORE` ■ `DISABLE-ACCOUNTING` ■ `DISABLE-CMDLOG-FILTER` ■ `DISABLE-CMDLOG` ■ `DISABLE-DYN-WORKER` ■ `DISCONNECT-PSTORE` ■ `ENABLE-ACCOUNTING` ■ `ENABLE-CMDLOG-FILTER`	R	Command to be performed. See ***List of Commands and Objects*** below.

Command-line Parameter	Option	Parameter	Req/ Opt	Explanation
		■ ENABLE-CMDLOG ■ ENABLE-DYN-WORKER ■ FORBID-NEWUOWMSGS ■ PING ■ PRODUCE-STATISTICS ■ PURGE ■ RESET-USER ■ RESUME ■ SET-CMDLOG-FILTER ■ SET-COLLECTOR ■ SET-UOW-STATUS ■ SHUTDOWN ■ START ■ STATUS ■ STOP ■ SUSPEND ■ SWITCH-CMDLOG ■ TRACE-FLUSH ■ TRACE-OFF ■ TRACE-ON ■ TRAP-ERROR		
object type	-d	■ BROKER ■ CONVERSATION ■ PARTICIPANT ■ PSF ■ SECURITY ■ SERVER ■ SERVICE ■ TRANSPORT	R	The object type to be operated on. See ***List of Commands and Objects*** below. Within EntireX Broker nomenclature, a participant is an application implicitly or explicitly logged on to the Broker as a specific user. See *Implicit Logon* and *Explicit Logon*. A participant could act as client or server.
	-D	collector brokerid	O	For command SET-COLLECTOR only. If provided, sets the collector ID to the given collector broker ID.
	-e	errornumber	O	Error number being trapped.

Command-line Parameter	Option	Parameter	Req/ Opt	Explanation
	-E		O	Exclude attach servers from service shutdown.
help	-h		O	Prints help information.
class/server/service	-n	class/server/service	O	Service triplet.
option	-o	■ ACCEPTED ■ CANCELLED ■ IMMED ■ QUIESCE ■ LEVEL*n*, where *n*=1-8	O	Command option.
puserid	-p	puserid	O	Physical User ID. For SERVER and PARTICIPANT objects only. This must be a hex value.
sslparms	-s	SSL parameters	O	When using SSL transport for broker communication. See ***Using SSL/TLS***.
seqno	-S	sequence number	O	Sequence number of participant.
token	-t	token	O	Token. For PARTICIPANT object only.
uowid	-u	uowid	O	Unit of work ID. For PSF object only.
userid	-U	userid	O	User ID. For PARTICIPANT object only.
secuserid	-x	userid	O	User ID for security purposes.
transportid	-X	Transport ID	O	One of the following: COM|NET|SSL|S*nn*|TCP|T*nn*. See table below.
secpassword	-y	password	O	Password for security purposes.
--encrypted_password_from_stdin	O	Encrypted password. See ***Using an Encrypted Password***.		

Transport ID Values

This table explains the possible values for parameter transportid:

Transport ID	Explanation
COM	all communicators
NET	NET transport communicator
SSL	all SSL communicators
S00	SSL communicator 1
S01	SSL communicator 2
S02	SSL communicator 3
S03	SSL communicator 4
S04	SSL communicator 5
TCP	all TCP/IP communicators
T00	TCP/IP communicator 1
T01	TCP/IP communicator 2
T02	TCP/IP communicator 3
T03	TCP/IP communicator 4
T04	TCP/IP communicator 5

Command-line Parameters from File

`etbcmd` supports an alternative method of passing command-line parameters.

If the environment variable `CMD_ATTR` is set, the content is interpreted as a file name. If no command-line parameters are given, the command `etbcmd` evaluates the content of the file. Example:

```
-blocalhost:3930:TCP
-cPRODUCE-STATISTICS
-dBROKER
```

List of Commands and Objects

This table lists the available commands and the objects to which they can be applied.

Command	Object							
	BROKER	CONVERSATION	PARTICIPANT	PSF	SECURITY	SERVER	SERVICE	TRANSPORT
ALLOW-NEWUOWMSGS				x				
APPMON-OFF	x							
APPMON-ON	x							
CLEAR-CMDLOG-FILTER	x							
CONNECT-PSTORE				x				
DISABLE-ACCOUNTING	x							
DISABLE-CMDLOG-FILTER	x							
DISABLE-CMDLOG	x							
DISABLE-DYN-WORKER	x							
DISCONNECT-PSTORE				x				
ENABLE-ACCOUNTING	x							
ENABLE-CMDLOG-FILTER	x							
ENABLE-CMDLOG	x							
ENABLE-DYN-WORKER	x							
FORBID-NEWUOWMSGS				x				
PING	x							
PRODUCE-STATISTICS	x							
PURGE				x				
RESET-USER					x			
RESUME								x
SET-CMDLOG-FILTER	x							
SET-COLLECTOR	x							

Command	Object							
	BROKER	CONVERSATION	PARTICIPANT	PSF	SECURITY	SERVER	SERVICE	TRANSPORT
SET-UOW-STATUS				x				
SHUTDOWN	x	x	x			x	x	
START								x
STATUS								x
STOP								x
SUSPEND								x
SWITCH-CMDLOG	x							
TRACE-FLUSH	x							
TRACE-OFF	x			x	x			
TRACE-ON	x			x	x			
TRAP-ERROR	x							

Note: Object type TRANSPORT applies to operating systems z/OS and z/VSE only.

Examples

Example	Description
etbcmd -b etb001 -h	Displays ETBCMD help text.
etbcmd -b etb001 -d BROKER -c TRACE-OFF	Turns Broker tracing off.
etbcmd -b etb001 -d BROKER -c TRACE-ON -o LEVEL2	Sets Broker trace level to 2.
etbcmd -b etb001 -d BROKER -c SHUTDOWN	Performs Broker shutdown.
etbcmd -b etb001 -d SERVICE -c SHUTDOWN -o IMMED -n ACLASS/ASERVER/ASERVICE	Shutdown service CLASS=ACLASS,SERVER=ASERVER,SERVICE=ASERVICE. See also SHUTDOWN SERVICE for more information on shutdown options.
	Create list of servers and shutdown specific server in two steps (first step uses etbinfo). See also SHUTDOWN SERVER.
etbinfo -b etb001 -d SERVER -l FULL -f"%USER-ID %SEQNO"	1. Determine a list of all servers with sequence numbers.
etbcmd -b etb001 -d SERVER -c SHUTDOWN -o IMMED -S32	2. Shutdown server with sequence number 32.
etbcmd -b etb001 -d BROKER -c PING	Performs an EntireX ping against the Broker.
etbcmd -b etb001 -d PSF -c DISCONNECT-PSTORE	Disconnects the Broker PSTORE.

Example	Description
`etbcmd -b etb001 -d PSF -c CONNECT-PSTORE`	Connects the Broker PSTORE.
`etbcmd -b etb001 -d PSF -c PURGE -u 100000000U00001A`	Purges a unit of work.
`etbcmd -b etb001 -d PSF -c ALLOW-NEWUOWMSGS`	Allows new units of work to be stored.
`etbcmd -b etb001 -d PSF -c FORBID-NEWUOWMSGS`	Disallows new units of work to be stored.
`etbcmd -b etb001 -d PSF -c SET-UOW-STATUS -o ACCEPTED -n ACLASS/ASERVER/ASERVICE`	Sets the status of UOWs that reside in the postpone queue back to `ACCEPTED` for service `ACLASS/ASERVER/ASERVICE`. See also *Postponing Units of Work* under *Using Persistence and Units of Work* in the Platform-independent Administration documentation.
`etbcmd -b etb001 -d PSF -c SET-UOW-STATUS -o CANCELLED -u 0010000000000100`	Cancel UOW with UOWID 0010000000000100 that resides in the postpone queue. See also *Postponing Units of Work*.

Using SSL/TLS

To set up SSL

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

2 Specify the Broker ID, using one of the following styles:

- *URL Style*, for example:

```
ssl://localhost:2010
```

- *Transport-method Style*, for example:

```
ETB024:1609:SSL
```

If no port number is specified, port 1958 is used as default.

3 Specify SSL parameters with the option -s|S (lowercase for `etbcmd`; uppercase for `etbinfo`). See *SSL/TLS Parameters for SSL Clients*.

4 Make sure the broker is prepared for SSL connections as well. See *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation.

Using an Encrypted Password

You can encrypt a password and store this in a file. Specify this file instead of a cleartext password when you call a secure broker.

Note: We strongly recommend that your cleartext password is longer than 16 characters.

To encrypt a password

1 Enter the command:

```
etbnattr --echo_password_only -w clear_text_password ↩
```

The encrypted password is written to stdout.

2 Copy the password value to an empty file. (Ignore the prefix `KEY-PASSWD-ENCRYPTED:`.)

To specify the encrypted password from stdin

- Enter the command:

```
etbcmd -xuid --encrypted_password_from_stdin < file
```

Where *file* is the file containing the encrypted password you created as described above. Example:

```
etbcmd -blocalhost:1971 -cPING -dBROKER -xUID --encrypted_password_from_stdin ↩
< myPwd
```

10 Attach Manager

EntireX includes an Attach Manager (ATM) for UNIX and Windows. This is used to start servers if a client requests a particular service from the Broker, but a server for that service is not active. This chapter covers the following topics:

Prerequisites

The Attach Manager needs the following:

- An active task registered at the Broker. As of EntireX 9.9, the ATM task is no longer launched automatically on each computer where EntireX is installed. See *Setting up the Attach Manager* for how to start the Attach Manager automatically or manually.
- A list of services the ATM is responsible for, and information on how to start the corresponding server for a particular service. The Attach Manager can start only processes that are local to where it is running, that is, the process that is attached will be run from the command line. There is no restriction, however, on what the started command-line process does, including starting a remote process on another system that will `REGISTER` as the server that satisfies the attach request.
- A configuration file that contains the service list the ATM is responsible for, information on how to start the corresponding server and additional configuration parameter to control the ATM functionality.

Setting up the Attach Manager

If you do not need the ATM for your own services, you do not need to perform any configuration for the ATM. For the default configuration EXXATM, a default configuration file *EXXATM.cfg* comes with the EntireX installation and contains the necessary configuration to start the EntireX sample servers. The file is located in directory *EntireX/config/service/appl.EXXATM*.

Notes:

1. In the current version of EntireX, the ATM is *not* launched automatically by default.
2. The command `etbsrv` uses the default section defined in the configuration file.

To launch the Attach Manager automatically

- Activate automatic start after a reboot or after a restart of the Broker Adminstrator Service with the following command:

```
etbsrv SERVICE ATTR <configuration name> AUTOSTART=YES
```

For example:

```
etbsrv SERVICE ATTR EXXATM AUTOSTART=YES
```

With the next reboot, ATM is then launched automatically. The working directory is *EntireX/config/service/appl.EXXATM*. All log files are written to this directory. It also contains the configuration file *EXXATM.cfg* of the Attach Manager. See ***Configuration File Syntax***.

To deactivate automatic start of the Attach Manager

- Enter command:

```
etbsrv SERVICE ATTR <configuration name> AUTOSTART=NO
```

To check the status of the Attach Manager

- Enter command:

```
etbsrv SERVICE STATUS <configuration name>
```

To start and stop the Attach Manager

- Enter one of the following commands:

```
etbsrv SERVICE START <configuration name>
```

```
etbsrv SERVICE RESTART <configuration name>
```

```
etbsrv SERVICE STOP <configuration name>
```

To show the current status

- Enter command:

```
etbsrv SERVICE STATUS
```

The Attach Manager is located in the *bin* subdirectory under the installed EntireX directory. The name of the executable is *exxatm.exe*. If you need to start an ATM manually for any reason, start it using this executable. Without further command-line arguments, the ATM uses the default section within the default configuration file. See *Operating the Attach Manager* for possible command-line arguments.

If you need multiple ATM instances, we recommend using a separate ATM configuration.

To create an Attach Manager configuration

- Enter the following command:

```
etbsrv SERVICE CREATE <configuration name>
```

Note: The created configuration is located in EntireX directory *config/service/appl.<configuration name>*

Configuration File Syntax

The syntax of the text-based configuration file is simple and is very similar to a Windows INI file.

Syntax Element	Description
;	Lines beginning with a semicolon are comment lines.
[ ]	Lines that contain text in square brackets are section headers.
Keyword=Value	Lines that are of the form Keyword=Value are keyword lines.

Note: Any of the values of the keywords in the configuration file can be set as environment variables.

There are three different types of sections in the configuration file:

- The ATM section to configure a particular ATM instance. The ATM section with the name "Default" is the default section. If no section with the name "Default" is found, the first ATM section in the file is the default section. Each ATM section contains the configuration parameters of the corresponding ATM instance and has one related Service List section, which refers to the services that this ATM supports. Each ATM section needs exactly one ATM server attaching the related servers if requested.
- The Service List section contains a list of names of Service sections. The name of the Service List section is the name of the related ATM section appended by "_Services".

- The Service section configures a service, which consists of the service name and how to start the corresponding server.

The general structure of the configuration file is the following:

```
[Default]
;  The parameters of the Default ATM
[Default_Services]
SERVICE1=
SERVICE2=
[SERVICE1]
;  The parameters for SERVICE1
[SERVICE2]
;  The parameters for SERVICE1
```


Parameters of the ATM Section

These sections define the Attach Manager itself and contain the keywords indicated in this table. There can be up to 16 of these sections.

Keyword	Definition and Value	Format	Example	Notes
BrokerID=	The Broker that ATM will communicate with and answer attach requests. Any valid ACI broker ID value is allowed.	A32	BrokerID= server1:1971:TCP	
SSLParms=	Secure Sockets Layer parameters for brokers that use SSL transport.	A512	SSLParms= VERIFY_ SERVER= N&TRUST_STORE= C:\\Temp \\ExxCACert.pem	
ServerClass= ServerName= Service=	The CLASS/SERVER/SERVICE names that can be used by ATM to send commands to ATM. The CLASS/SERVER/SERVICE name needs to be defined in the *Broker Attributes*.	A32 [for all keywords]	ServerClass= System ServerName= DefaultMain Service= Command	
UserID=	The user ID of the ATM.	A32	UserID=atman	

Keyword	Definition and Value	Format	Example	Notes
Token=	The token of the ATM (used for unique identification of the user ID). There is a special value of {GeneratedToken} which will generate a unique 32-byte value for the ATM.	A32	Token=atm Token={GeneratedToken}	
Password=	Password for the user ID.	A32	Password=atman	
PwdEncrypted=	Encrypted password for the user ID. If keyword PwdEncrypted is specified, keyword Password (containing the clear text password) can be omitted. If both keywords (PwdEncrypted and Password) are specified, the value PwdEncrypted is used.	A256	PwdEncrypted=1B6C607...	You can generate the encrypted password with command etbnattr ↩ --echo_password_only ↩ -w clear_text_passwor
WaitTime=	During the specified time, the Attach Manager waits for a response. After expiration of the time, the Attach Manager receives a timeout. This is used as the WAIT time on the ATM's RECEIVE call.	A8	WaitTime=5M	Identical to Broker control block WAIT parameter. Default=60S.
RecvLength=	Size of the buffer that is available for receiving orders.	I4	RecvLength=12000	Identical to Broker control block RECEIVE-LENGTH parameter. Default=8000.
HistoryFile=	File name for logging orders that have been received for restarting. If this keyword is not specified, no file is written. This can be any valid file name.	Valid path name for dependent platform. See example.	HistoryFile=%TEMP%\Default.his	
HistoryFileMode=	When starting the Attach Manager, you can decide here whether the current file is to be overwritten or not.	w or a+t	HistoryFileMode=w	File is newly opened for writing; the old file is deleted.

Keyword	Definition and Value	Format	Example	Notes
			HistoryFileMode=a+t	Writing of the old continued.
LogFile=	Log information is logged here about the current status of the Attach Manager. If this keyword is not specified, no file is written.	Valid path name for dependent platform. See example.	LogFile=%TEMP%\Default1.log	
DailyLogFile	Split LogFile on a daily basis.	Y	DailyLogFile=Y	If more than one sp is specified, the fo logic is used: 1. daily 2. monthly 3. by size
MonthlyLogFile	Split LogFile on a monthly basis.	Y	MonthlyLogFile=Y	
MaxSizeLogFile	Split LogFile based on the configured file size (KB/MB/GB/TB/PB).	A32	MaxSizeLogFile=16GB or MaxSizeLogFile=10000	
MaxTraceFiles	Maximum number of backup files.	I4	MaxTraceFiles=3	Default=0
LogFileMode=	When starting the Attach Manager, the administrator can decide whether the current file is to be overwritten or not. The file can get very large.	w or a+t	LogFileMode=w	File is newly open writing; the old fil deleted.
			LogFileMode=a+t	Writing of the old continued.
Sleep=	If the Attach Manager cannot register successfully during startup, or if a connection is broken, the Attach Manager waits this specified time in seconds and then tries again. You can limit the number of connection attempts, using the keyword Retries=*n*.	I4	Sleep=120	
Retries=	If registration fails, the number of subsequent registration attempts can be limited. the keyword Sleep determines the wait time before a new registration attempt. Setting Retries=0 deactivates this functionality.	I4	Retries=0	Default=0.

Keyword	Definition and Value	Format	Example	Notes
ShutdownBy UserRequest=	When set to 1, the ATM can be stopped when a command is sent to it to shut down. If it is set to zero, it will restart automatically.		Values: 0 Attach Manager restarts. The configuration file is read anew. 1 Attach Manager terminates itself.	

Parameters of the Service List Section

This section names the Service sections that will be used to define the services that will be attached. The prefix of the name of the section must match a specific instance of the AttachManager(n) sections.

Example: Assume there are three services to be attached. They can be logically defined as follows:

```
[Default_Services]
payroll=
inventory=
qualitycontrol=
```

Therefore, there will be three optional sections following: `[payroll]`, `[inventory]`, and `[qualitycontrol]`.

Parameters of the Service Section

There can be any number of Service sections attached to an ATM by means of its corresponding Service List section. The Service sections are used to define the actual commands that will be issued by ATM to attach servers to respond to Broker requests

The following keywords can be used:

Keyword	Definition	Format	Example
ServerClass= ServerName= Service=	The CLASS/SERVER/SERVICE name of the service to be attached.	A32	ServerClass=ACLASS ServerName=ASERVER Service=ASERVICE
Min=	The minimum number of servers that should be active. Value must be greater than 0.	I4	Min=3
Max=	The maximum number of servers that should be active.	I4	Max=7
Increment=	The number that should be started when a request is made, up to the number indicated by Max=.	I4	Increment=1

Keyword	Definition	Format	Example
`Command=`	Command-line command to be issued that will start the service.	Specifies (a) the fully qualified path to the location of the executable to be run and (b) the options for that executable. See example.	`Command=./server/myserver.exe`

Example from table above: If there are no instances of the service `ACLASS:ASERVER;ASERVICE REGISTERED`, the command indicated in the `Command=` keyword will be issued three times.

Sample Configuration File

Note: A sample configuration file is provided in the */config* directory of EntireX. This sample defines two ATMs: Default and AttachManager2. The default ATM supports only the services related to Default.

```
[Default]
;
; Specify the broker to which the Attach Manager attaches and
; the channel on which the Attach Manager listens for command
; requests.
;
BrokerID=localhost:1971:TCP
ServerClass=System
ServerName=DefaultMain
Service=Command
UserID=%USERNAME%
Token={GeneratedToken}
Password=Hugo
WaitTime=30s
RecvLength=12000

; Activities will be written to the history file (optional)
HistoryFile=%TEMP%\Default.his
HistoryFileMode=a+t

; Log messages will be written to the log file (optional)
LogFile=%TEMP%\Default.log
; Append to an existing file
;LogFileMode=a+t
; Create a new file
LogFileMode=w

Sleep=10
```

```
Retries=0

ShutdownByUserRequest=1

;
;
; Default's services
;
[Default_Services]
AServer=
BServer=
;
[AServer]
ServerClass=ACLASS
ServerName=ASERVER
Service=ASERVICE
Min=2
Max=3
Increment=1
Command=myservera -c<ServerClass> -s<ServerName> -v<Service> -b<BrokerID> -i500
;
[BServer]
ServerClass=BCLASS
ServerName=BSERVER
Service=BSERVICE
Min=1
Max=1
Increment=1
Command=myserverb -c<ServerClass> -s<ServerName> -v<Service> -b<BrokerID> -i750
[AttachManager2]
;
; Specify the broker to which the Attach Manager attaches and
; the channel on which the Attach Manager listens for command
; requests.
;
BrokerID=localhost:1971:TCP
ServerClass=System
ServerName=AttachManager2Main
Service=Command
UserID=%USERNAME%
Token={GeneratedToken}
Password=Hugo
WaitTime=30s
RecvLength=12000

; Activities will be written to the history file (optional)
HistoryFile=%TEMP%\AttachManager2.his
HistoryFileMode=a+t

; Log messages will be written to the log file (optional)
LogFile=%TEMP%\AttachManager2.log
; Append to an existing file
```

```
;LogFileMode=a+t
; Create a new file
LogFileMode=w

Sleep=10

ShutdownByUserRequest=1
;
; AttachManager2's services
;
[AttachManager2_Services]
CServer=
;
[CServer]
ServerClass=CCLASS
ServerName=CSERVER
Service=CSERVICE
Min=1
Max=1
Increment=1
Command=myserverc -c<ServerClass> -s<ServerName> -v<Service> -b<BrokerID> -i1000
```

Operating the Attach Manager

Under normal circumstances, no manual operation is not necessary if the default ATM satisfies your needs. However, if you need to run multiple ATMs in your environment, this section describes how to perform the necessary operations.

Starting the Attach Manager

To start an Attach Manager

- Enter command:

```
etbsrv SERVICE <configuration name>
```

Note: `etbsrv` starts with the default section defined in the configuration file.

Or:

Either from the *bin* directory of EntireX (or from any directory if the *bin* directory is in the PATH), enter the following command:

```
exxatm -F<full-path of Configuration file> -N<AttachManager1> -N<AttachManager2> ↩
...
```

Notes:

1. With the `-N` argument you specify the ATM section for which the Attach Manager is responsible for. If this argument is omitted the attach manager is responsible for the default section.
2. With the `-F` argument you specify the location of the configuration file. If this argument is omitted, the Attach Manager uses the default configuration file. All ATM instances should use the same configuration file, so we recommend you use the default file for the default ATM.
3. The Attach Manager writes output to stdout. If you start the Attach Manager as a background process, stdout must be redirected to a file.

Stopping the Attach Manager

To stop an Attach Manager

Each attach manager corresponds to an particular broker and registers a command service defined with the configuration variables ServerClass/ServerName/Service in the ATM section.

- Use the script `etbsrv`.

  Or:

  Use the command-line utility `etbcmd`.

  Or:

  Press **CTRL-C**.

  Or:

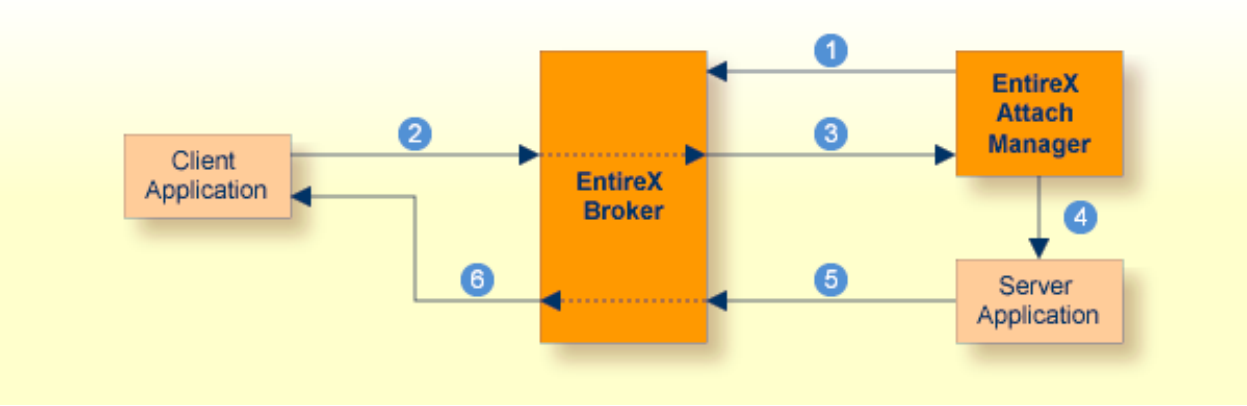
  Under UNIX, enter command `kill process-id`.

Logging the Attach Manager

The ATM log file and a history file are defined by the ATM configuration parameters. For each order to launch a service, the ATM writes a record into the history file. The history record has the following format:

- date and time
- the service name as defined in the ATM config file
- server name, server class and service
- number of active replicates (this number is greater than 0 only if all running replicates are busy while a new client requests the service
- number of server lookups, that is, the number of clients requesting a new replicate of the server; this is greater than 1 only if two clients request a service in parallel
- replicate increment as defined in the ATM config file
- number of replicates actually launched; this differs from the increment only if the high watermark is exceeded

Attach Manager Processing



1. Attach Manager registers with Broker, indicating that it will attach named services. These are called attach-managed services.
2. Client requests a service that is attach-managed. Server may or may not be active. If it is not, a server will be started (attached).
3. Attach request comes from the Broker.
4. Attach Manager issues command to start the server application.
5. Server application issues a `LOGON` to the Broker, then issues `REGISTER` and `RECEIVE`. It gets message from client, processes the message, and responds.
6. Response from server is received by the client application.

11 Settting up and Administering the EntireX Broker TCP Agent

The EntireX Broker TCP Agent is a gateway to the broker whenever direct TCP/IP communication with the broker is not possible. Under UNIX, use the delivered script */opt/softwareag/EntireX/bin/brokeragent.bsh* to start the agent.

Common Scenarios

The most common scenarios for using the Broker TCP Agent are where the Java security manager does not allow direct communication with the Broker. For example, an untrusted Java applet can only open a TCP/IP connection to a Broker which is running on the same machine as the Web server.

Although in most cases the Broker TCP Agent will be used from a Broker application written in Java, the Broker TCP Agent can also be used from any component or application configured with TCP/IP.

Indirect TCP/IP Connections by the TCP Agent to Avoid Security Restrictions

The Broker TCP Agent must be used when the Java client cannot open a TCP/IP connection to the EntireX Broker due to security or firewall settings. The most prominent case is the Java sandbox model, which permits a Java applet to open only TCP/IP connections to the machine where the Web server resides. If the EntireX Broker is running on a different machine, a TCP Agent has to be run on the Web server machine.

Using the TCP Agent

Class Name and Parameters

The Broker TCP Agent is a standalone Java application. The class name which contains the `main` method is `com.softwareag.entirex.ba.BrokerAgent`.

Specify the following parameters in the order given in this table when the TCP Agent listens on a TCP/IP port:

Parameter	Explanation
1. Trace Option	Valid values: ON or OFF. Default: OFF. A dump of the buffers is written to standard output for diagnostic purposes.
2. Port Number	The port number the TCP Agent uses for incoming requests from Broker applications. This port number must be specified as part of the Broker ID in the Broker application.
3. Broker Address	The TCP Agent sends all requests to this Broker using any legal Broker ID defined with *URL-style Broker ID*. The TCP Agent will use direct TCP/IP communication if the TCP/IP protocol is used (the address is of the form `Hostname`, `Hostname:Number` or starts with `tcpip://`).
4. Bind Address	The address of the network interface on which the Broker TCP Agent will listen for connection requests. The default is that the Broker Agent will listen on any attached interface adapter of the system. The bind address is the local IP address or host name to bind to.

Starting the TCP Agent

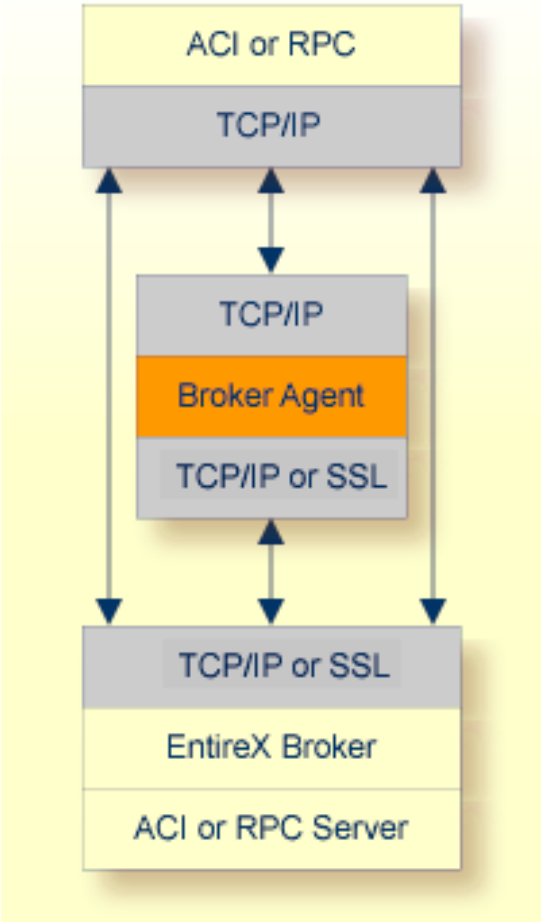
Under UNIX, the EntireX distribution kit comes with a shell script to start the Broker TCP Agent. Change the port number and the Broker address in the script /<*Install_Dir*>/*EntireX/bin/broker-agent.bsh*.

Activating Tracing for the TCP Agent

Set the parameter Trace Option to "ON". See ***Class Name and Parameters***.

Architecture of the Broker TCP Agent

The architecture of the Broker TCP Agent is shown in the following picture:



12 Settting up and Administering the EntireX Broker SSL Agent

The EntireX Broker SSL Agent is a gateway to the broker whenever direct SSL/TLS communication with the broker is not possible. Under UNIX, use the delivered script */opt/softwareag/EntireX/bin/sslbrokeragent.bsh* to start the agent.

Common Scenarios

The most common scenarios for using the Broker SSL Agent are where direct SSL communication to the Broker is not possible or it is not required by the network architecture.

Although in most cases the Broker SSL Agent will be used from a Broker application written in Java, the Broker SSL Agent can also be used from any component or application configured with SSL. See *Using SSL/TLS with EntireX Components*.

Using the Broker SSL Agent

Class Name and Parameters

The Broker SSL Agent is a standalone Java application. The class name is `com.softwareag.entirex.ba.SSLBrokerAgent`.

Specify the following parameters in the order given in this table when the Broker SSL Agent listens on an SSL port:

Parameter	Explanation
1. Trace Option	Valid values: ON or OFF. Default: OFF. A dump of the buffers is written to standard output for diagnostic purposes.
2. Port Number	The port number the Broker TCP Agent uses for incoming requests from Broker applications. Specify this port number as part of the broker ID in the broker application.
3. SSL Parameters	SSL parameters when the Broker SSL Agent runs as an SSL server. SSL requires a (server) certificate with a private key. Specify with `key_store=filename` the file name of a Java keystore that contains the private key. SSL client authentication can be requested with the parameter `verify_client=yes`. In this case, specify with `trust_store=filename` the file name of a Java keystore containing the list of trusted certificate authorities that issued the client's certificate. The complete list of parameters could be `key_store=keystore&verify_client=yes&trust_store=castore`. Examples: `key_store=ExxJavaAppCert.jks trust_store=ExxCACert.jks.` See also *SSL/TLS Parameters for Broker as SSL Server (One-way SSL)*.
4. Password	The password which protects the private key. If the value `-prompt` is specified the password is read from standard input.

Parameter	Explanation
5. Broker Address	The Broker SSL Agent sends all requests to this Broker using any legal Broker ID defined with *URL-style Broker ID*. The Broker SSL Agent will use SSL communication if the SSL protocol is used (the address starts with `ssl://`).
6. Bind Address	The address of the network interface on which the Broker SSL Agent will listen for connection requests. The default is that the Broker Agent will listen on any attached interface adapter of the system. The bind address is the local IP address or host name to bind to.

Starting the Broker SSL Agent

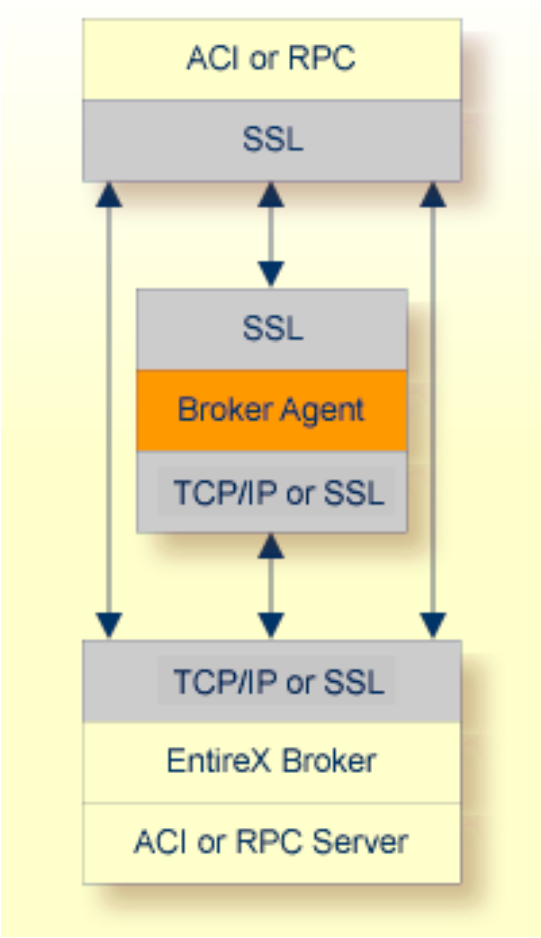
Under UNIX, the EntireX distribution kit comes with a shell script to start the Broker SSL Agent. Change the port number, the Broker address and the SSL parameters in script */<Install_Dir>/EntireX/bin/sslbrokeragent.bsh*.

Activating Tracing for the Broker SSL Agent

Set the parameter Trace Option to "ON". See ***Class Name and Parameters***.

Architecture of the Broker SSL Agent

The architecture of the Broker SSL Agent is shown in the following picture:


ACI or RPC
SSL
SSL
Broker Agent
TCP/IP or SSL
TCP/IP or SSL
EntireX Broker
ACI or RPC Server

13 Settting up and Administering the EntireX Broker HTTP(S) Agent

The EntireX Broker HTTP(S) Agent is a Java-based component that implements a Java servlet for servlet-enabled Web servers. It builds the bridge between a Web server and EntireX Broker in the intranet.

HTTP(S) Tunneling with EntireX

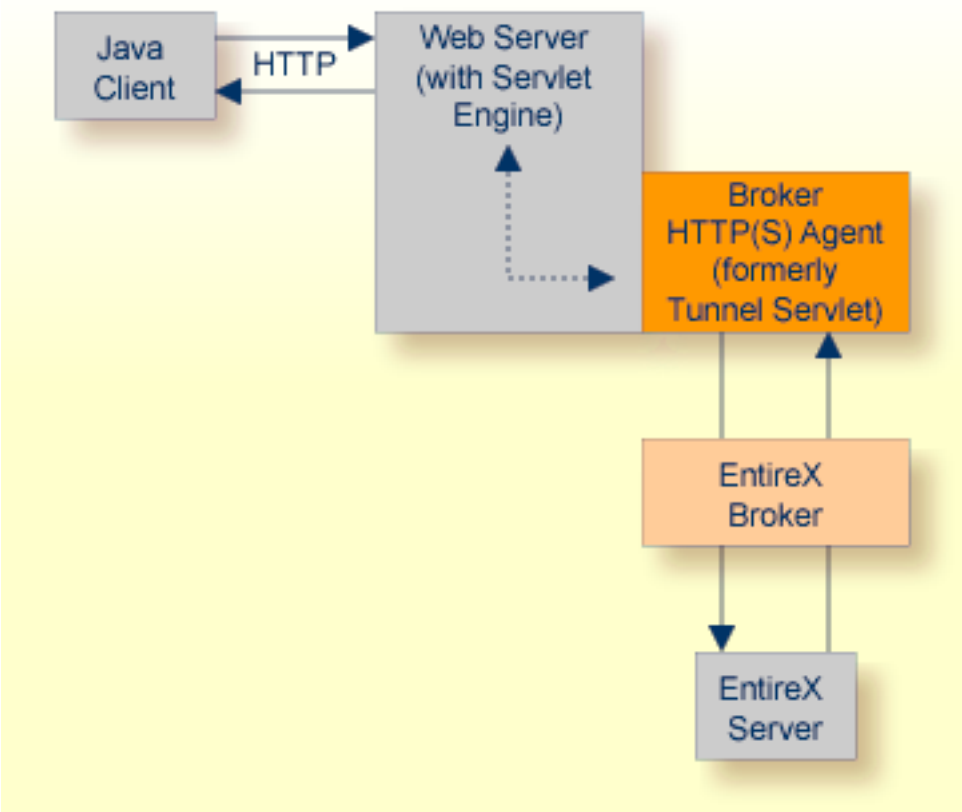
Introduction

When communicating with EntireX Broker over the internet, direct access to the EntireX Broker's TCP/IP port is necessary. This access is often restricted by proxy servers or firewalls. With EntireX, Java-based communication components can pass communication data via HTTP or HTTPS. This means a running EntireX Broker in the intranet is made accessible by a Web server without having the need to open additional TCP/IP ports on your firewall (HTTP tunneling). HTTP or HTTPS tunneling can also be used for Java RPC.

How the Communication Works

The EntireX Java ACI is able to send and receive data via an HTTP protocol controlled by constructor `com.softwareag.entirex.aci.Broker`. See *How to Enable HTTP Support in a Java Component* under *Writing Advanced Applications - EntireX Java ACI*.

The EntireX Java component `com.softwareag.entirex.aci.TunnelServlet.class` implements a Java servlet for servlet-enabled Web servers. It builds the bridge between Web server and EntireX Broker in the intranet.



The figure above shows how the communication works. In this scenario, a Java client program communicates via HTTP and EntireX Broker with an EntireX server. By using a Broker ID starting with http:// (passing the URL of the installed Broker HTTP(S) Agent) each Broker request is sent to a Web server, which immediately processes the Broker HTTP(S) Agent, passes the contents to EntireX Broker, receives the response and sends it back via HTTP. For the two partners (client and server) it is transparent that they are communicating through the Web. Java server programs can also communicate via HTTP if necessary.

Configuring the Broker HTTP(S) Agent

To use the Broker HTTP(S) Agent you need a servlet-enabled Web server.

Parameter	Description	
broker	The broker you want to address (syntax: as Broker ID in Java).	
log	**Yes**	Default. Servlet writes logging information to its standard output.
	No	No log is created.

In the following, “tunnel” is used as the agent name.

To adapt the Broker HTTP(S) Agent

The following steps describe the deployment with the Web archive *entirex.jar* in detail. You can test the Broker HTTP(S) Agent with `http://<host>:<port>/entirex/tunnel`, where `entirex` is the name of the Web application.

1 Create the new subfolders in the Web application directory of your Web server, e.g. *tunnel*, *tunnel/WEB-INF*, *tunnel/WEB-INF/lib*.

2 Copy the *entirex.jar* into *tunnel/WEB-INF/lib*.

3 Create a file named *web.xml* in the folder *tunnel/WEB-INF* with the following content:

```
<web-app>
   <servlet>
     <servlet-name>tunnel</servlet-name>
     <servlet-class>com.softwareag.entirex.aci.TunnelServlet</servlet-class>
     <init-param>
       <param-name>broker</param-name>
       <param-value>yourbroker</param-value>
     </init-param>
     <init-param>
       <param-name>log</param-name>
       <param-value>yes</param-value>
     </init-param>
   </servlet>
   <servlet-mapping>
     <servlet-name>tunnel</servlet-name>
     <url-pattern>/*</url-pattern>
   </servlet-mapping>
 </web-app>
```

4 Restart your Web server and test the installation by calling the Broker HTTP(S) Agent in your Web browser. The URL is: *http://<yourhost>/tunnel*. If the agent is installed properly, an information page is displayed.

5 Run either the RPC `CALC` example or the `bcoc`/`bcos` broker verification.

- To run the RPC `CALC` example, see the relevant section for Natural | COBOL | PL/I and also *EntireX IDL Tester* in the Designer documentation.
- To use the `bcoc`/`bcos` verification, see *Sample Programs for Client (`bcoc`) and Server (`bcos`)* in the z/OS | UNIX | Windows | BS2000 installation documentation or *Verifying the Installation of the EntireX Broker* (z/VSE | BS2000).

Using Internationalization with the Broker HTTP(S) Agent

Character conversion is transparent for the Broker HTTP(S) Agent. The client sending the EntireX ACI request with HTTP over the Web server through the Broker HTTP(S) Agent fully controls its encoding. No configuration is necessary for the Broker HTTP(S) Agent.

Activating Tracing for the Broker HTTP(S) Agent

To switch on tracing for the Broker HTTP(S) Agent

- Set the system property `entirex.trace` to one of the values 1, 2, or 3. See *Tracing*.

To switch on logging

- Set the configuration parameter `log=yes`.

  This logs the parameters from the HTTP header, the HTTP messages and error messages to the logging facility of the Web server.

14 Tracing webMethods EntireX

This chapter describes the various techniques available for troubleshooting, tracing and logging with EntireX components.

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

Table Summarizing Tracing for webMethods EntireX Components

EntireX Component	Use Tracing Technique for	Tracing Technique
Broker ActiveX Control	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing Broker Stubs*
EntireX Broker ACI under Windows	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing Broker Stubs*
EntireX Broker Agent	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing Broker Agent*
EntireX Broker under UNIX	Processing within the Broker Requests to, replies from clients/server	*Tracing EntireX Broker*
DCOM Wrapper	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing Broker Stubs*
	RPC-related problems on the client side Requests to, replies from RPC Servers Requests to, replies from the Broker	*Tracing the RPC Runtime*
EntireX Java ACI	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing EntireX Java ACI*
Java Wrapper	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing EntireX Java ACI*
EntireX RPC Server for Java	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing RPC Server for Java*
EntireX IDL Tester		
.NET Wrapper	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing Broker Stubs*
	RPC-related problems on the client side Requests to, replies from RPC servers Requests to, replies from the Broker	*Tracing the RPC Runtime*
C Wrapper	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing Broker Stubs*

EntireX Component	Use Tracing Technique for	Tracing Technique
	RPC-related problems on the client side Requests to, replies from RPC servers Requests to, replies from the Broker	*Tracing the RPC Runtime*
RPC Server	RPC-related problems on the server side Requests to, replies from RPC clients Requests to, replies from the Broker	*Activating Tracing for the RPC Server* for C | .NET | Micro Focus
	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing Broker Stubs*
EntireX Broker HTTP(S) Agent	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing EntireX Java ACI*
EntireX RPC Server for XML/SOAP	For RPC Server for XML/SOAP-related problems.	*Tracing the XML/SOAP Runtime*
	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing EntireX Java ACI*
EntireX XML Tester		
EntireX Listener for XML/SOAP	For Listener for XML/SOAP-related problems.	*Tracing the XML/SOAP Runtime*
	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing EntireX Java ACI*
XML/SOAP Wrapper	For XML/SOAP Wrapper-related problems.	*Tracing the XML/SOAP Runtime*
	Transport-related problems Requests to, replies from the Broker or Broker Agent	*Tracing EntireX Java ACI*
EntireX RPC-ACI Bridge		*Tracing the EntireX RPC-ACI Bridge*

Tracing EntireX Broker

See also *EntireX Broker Return Codes*.

Switching on Tracing

To switch on tracing

- Set the attribute `TRACE-LEVEL` in the broker attribute file
  - for minimal trace output to "1"
  - for detailed trace output to "2"
  - for full trace output to "3"

  Example:

```
TRACE-LEVEL=2
```

Switching off Tracing

To switch off tracing

- Set the attribute `TRACE-LEVEL` in the broker attribute file to 0:

```
TRACE-LEVEL=0
```

  Or:

  Omit the `TRACE-LEVEL` attribute.

Viewing the Trace Log

The trace file, *`BrokerID`.LOG*, is written to the *Broker Directory*.

To view the contents of a log

- Using Command Central, select an environment in the **Environments** pane, select the **Instances** tab, click the name of a product instance, select the **Logs** tab, click the log alias for a log in the **Alias** column.

  Or:

  Enter the following command in Command Central:

```
sagcc get diagnostics logs
```

This retrieves log entries from a log file. Log information includes the date, time, and description of events that occurred with a specified runtime component.
See *Administering EntireX Components with Command Central* in the EntireX documentation or the separate Command Central documentation and online help for details.

Deferred Tracing

It is not always convenient to run with `TRACE-LEVEL` defined, especially when higher trace levels are involved. Deferred tracing is triggered when a specific condition occurs, such as an ACI response code or a broker subtask abend. Such conditions cause the contents of the trace buffer to be written, showing trace information leading up the specified event. If the specified event does not occur, the Broker trace will contain only startup and shutdown information (equivalent to `TRACE-LEVEL=0`). Operating the trace in this mode requires the following additional attributes in the broker section of the attribute file. Values for `TRBUFNUM` and `TRAP-ERROR` are only examples.

Attribute	Value	Description
`TRBUFNUM`	`3`	Specifies the deferred trace buffer size = 3 * 64 K.
`TRMODE`	`WRAP`	Indicates trace is not written until an event occurs.
`TRAP-ERROR`	`322`	Assigns the event ACI response code 00780322 "PSI: UPDATE failed".

Dynamically Switching On or Off the EntireX Broker Trace

The following methods are available to switch on or off the EntireX Broker trace dynamically. You do not need to restart the broker for the changes to take effect.

- `etbcmd`
  Run command utility `etbcmd` with option `-c TRACE-ON or - c TRACE-OFF`. See `etbcmd`.
- **Command Central**
  Use Command Central. See *Updating the Trace Level* under *Administering the EntireX Broker* using the Command Central GUI | Command Line.

Tracing Broker Agent

To switch on tracing

- Set the parameter Trace Option to ON. For the complete table of parameters, see ***Using the Broker SSL Agent*** and ***Using the TCP Agent***.

To switch off tracing

- Set the parameter Trace Option to OFF.

  Or:

  Omit the parameter Trace Option.

Trace Output

The trace output is written to STDOUT.

Tracing Broker Stubs

The broker stubs provide an option for writing trace files.

To switch on tracing for the broker stub

- Before starting the client application, set the environment variable `ETB_STUBLOG`:

Trace Value	Trace Level	Description
0	`NONE`	No tracing.
1	`STANDARD`	Traces initialization, errors, and all ACI request/reply strings.
2	`ADVANCED`	Used primarily by system engineers, traces everything from level 1 and provides additional information - for example the Broker ACI control block - as well as transport information.
3	`SUPPORT`	This is full tracing through the stub, including detailed traces of control blocks, message information, etc.

Example:

```
ETB_STUBLOG=2
```

If the trace level is greater than 1, unencrypted contents of the send/receive buffers may be exposed in the trace.

The trace file is created in the current directory. The name is *`pid`.etb* where *`pid`* is the process ID. If you want to write the trace file to a different location, set environment variable `ETB_STUBLOGPATH` to the desired path.

See also ***UNIX Commands to Set the Environment Variables***.

Remember to switch off tracing to prevent trace files from filling up your disk.

To switch off tracing for the broker stub

- Set the environment variable `ETB_STUBLOG` to `NONE` or delete it.

Tracing EntireX Java ACI

The EntireX Java ACI provides a system property for tracing.

To switch on tracing

1 When starting the Java virtual machine, set the Java system property `entirex.trace`

   - for minimal trace output to "1"
   - for detailed trace output to "2"
   - for full trace output to "3".

2 The programming interface of the EntireX Java ACI allows you to set the trace value by the Java application using the EntireX Java ACI, see *Tracing* under *Writing Advanced Applications - EntireX Java ACI*. There may also be other methods to provide the trace value. See your application documentation.

To switch off tracing

- Set the Java system property `entirex.trace` to 0 when starting the Java virtual machine

  Or:

  Omit the Java system property `entirex.trace` when starting the Java virtual machine.

Trace Output

The trace output will be written to STDOUT.

Tracing RPC Server for Java

To switch on tracing

- When starting the Java virtual machine, set the Java system property `entirex.trace`
  - for minimal trace output to "1"
  - for detailed trace output to "2"
  - for full trace output to "3".

  See *Customizing the RPC Server*.

To switch off tracing

- Set the Java system property `entirex.trace` to "0" when starting the Java virtual machine.

  Or:

  Omit the Java system property `entirex.trace` when starting the Java virtual machine.

Trace Output

The trace output will be written to STDOUT.

Tracing the RPC Runtime

To switch on tracing

- Before starting the client application, set the environment variable `ERX_TRACELEVEL` to
  - `STANDARD` for minimal trace output
  - `ADVANCED` for detailed trace output
  - `SUPPORT` for full trace output.

To switch off tracing

- Set the environment variable to NONE or delete it.

Trace Output

By default the trace file, *ERXTrace.nnn.log*, will be written to the trace directory.

The value *nnn* can be in the range from 001 to 005.

To change the trace destination

- Set the environment variable `ERX_TRACEFILE` to the desired destination, which can consist of

  file names, folder names and variables for file names, folder names, process ID, thread ID, range.

  The variables are:

Variable	Operating System	Description
%...%	Windows	environment variable
$(...)	UNIX	environment variable
@PID	UNIX, Windows	process ID
@TID	UNIX, Windows	thread ID
@RANGE[*n*,*m*]	UNIX, Windows	*m* must be greater than *n*, range is from 0 - 999
@CSIDL_PERSONAL	Windows	The user's home directory. The variable will be resolved by Windows shell functions.
@CSIDL_APPDATA	Windows	The *Application Data Directory*. The variable will be resolved by Windows shell functions.
@CSIDL_LOCAL_APPDATA	Windows	The *Application Data Directory*. The variable will be resolved by Windows shell functions.

Related Information

Environment Variables in EntireX

Tracing the XML/SOAP Runtime

This section provides information on tracing the following components:

- EntireX RPC Server for XML/SOAP
- EntireX Listener for XML/SOAP
- EntireX XML/SOAP Wrapper

The following topics are covered:

Enabling Tracing

There are two ways to switch on tracing mode:

Using a Property File

To switch on tracing mode using a property file

1 Copy the trace property file *entirex.trace.standard* to one of the following locations:

- the working directory of your client application;
- the user's home directory;
- any other location.

2 Rename the copied file *entirex.trace.properties*.

3 Customize *entirex.trace.properties* as described in ***Trace Parameters***.

4 If *entirex.trace.properties* is within the current directory of your client application or your user home directory, it will be located automatically.

Otherwise, specify the fully qualified or relative file name when starting the Java virtual machine for your client application using property `entirex.sdk.default.trace.propertiesfile`, example:

```
java -Dentirex.sdk.default.trace.propertiesfile ↩
="/MyDirName/entirex.trace.properties" MyClient
```

Using Trace Parameters of the Java Virtual Machine

To switch on tracing mode by specifying the trace parameters of the Java virtual machine

- Submit the trace parameters when you start the Java virtual machine for the application to be traced. See *Trace Parameters*. Note that parameter specifications submitted overwrite settings in the property file.

Disabling Tracing

To switch off tracing

- Delete or rename the trace property file if it is located in the working directory or in the user's home directory.

  Or:

  Specify `level=NONE` when invoking the Java virtual machine :

```
java -Dentirex.sdk.default.trace.level = NONE MyClient
```

Configuring a Trace File for the Listener for XML/SOAP

We recommend to add the following parameter in file *conf/axis2.xml* located in the Software AG Common Web Services Stack installation:

```
<parameter name="exx-trace-propertiesfile">file:////path of trace.properties ↩
file</parameter>
```

Example:

```
<parameter ↩
name="exx-trace-propertiesfile">MyDirName/entirex.trace.properties</parameter>
```

Notes:

1. If a relative path is specified, the file is located in directory *WEB-INF/conf/* in the Web Services Stack web application file that contains the property.
2. In the parameter section of the file *axis2.xml*, the value of the parameter `exx-trace-propertiesfile` can contain definitions of operating system variables, for example `location="$EXXDIR/config/entirex.trace.properties"`.

Configuring a Trace File for the XML/SOAP Wrapper or the RPC Server for XML/SOAP

See *Enabling Tracing*.

Note: If the RPC Server for XML/SOAP is running as a daemon, enable tracing by adding a Java system property to the start script or by copying file `entirex.trace.properties` to the same directory as the start script.

Trace Parameters

The following table provides an overview of trace parameters, their respective values, and how to submit them as arguments when invoking the Java virtual machine for the component to be traced.

Parameter	Syntax	Description
propertiesfile	entirex.sdk.*component name*.trace.propertiesfile= *absolute or relative path including the properties file*	Provide the location of the *entirex.trace.properties* file. Only used when the component is started. **Note:** A sample trace property file named *entirex.trace.standard* with predefined trace settings is contained in the directory *../EntireX/config*. This file is a model and must be renamed to the valid name when used.
level	entirex.sdk.*component name*.trace.level = *tracelevel*	You can specify the following trace levels: Keyword / Level / Description NONE / No tracing / Tracing is switch off STANDARD / User / Trace invocation of a component. ADVANCED / Expert / For support and diagnostics. Expert knowledge of the component is required. SUPPORT / Expert / Full trace output. Otherwise, as above.
directory	entirex.sdk.*component name*.trace.directory = *absolute or relative path*	Default is the working directory.
filename	entirex.sdk.*component name*.trace.filename = *FILE|STDOUT|STDERR*	Specify where tracing data is written to: Keyword / Destination STDOUT (Default) / Console STDERR / Console

Parameter	Syntax	Description
		FILE — File name is generated internally: *exx.sdk.component name.threadName.backupNo.log* , where backupNo is in the range from ".000" to ".009". Note that the number of files created depends on *maximumsize*. If more than 10 files are required, the oldest backup file is overwritten.
threadoriented	entirex.sdk.*component name*.trace.threadoriented = true | false	Keyword — Description YES — Thread-oriented: trace data is distributed over multiple files (one file per thread) NO (Default) — Trace data is written to one file.
rowlength	entirex.sdk.*component name*.trace.rowlength = *maximum_characters_per_row*	Maximum number of characters per row. If this limit is exceeded, the remaining letters are written to a new line.
maximumsize	entirex.sdk.*component name*.trace.maximumsize = *max_file_size*	Maximum size of the log file. If this limit is exceeded, the log file is renamed and the remaining data is written to a new log file, see *filename*. Note that this specification has an effect only if filename is set to "FILE".
timeframe	entirex.sdk.*component name*.trace.timeframe = *number of day*	Time period after which the log file is closed. If this time limit has exceeded, the log file is renamed and the remaining data (if any) is written to a new log file. Note that this specification has an effect only if filename is set to "FILE". You can specify the following timeframes: Keyword — Description 1-9+H — Number of hours 1-9+D — Number of days If no time frame is defined, only one log file is used until tracing is stopped. Example: If timeframe has been set to 30D, the current log file is closed and renamed at midnight every thirty days, and tracing is continued with a new log file.

Component Names

Trace properties given in the trace property file might have to be restricted by *componentname*. The following components are available:

EntireX Component	componentname	Description
	default	The trace property is not restricted to a specific EntireX component.
XML/SOAP Runtime	xml.runtime	The trace property belongs to the EntireX XML/SOAP Runtime only.

Tracing the EntireX RPC-ACI Bridge

To trace Broker calls

1 Use the system property `entirex.trace=[0|1|2|3]`.

This trace does not separate the calls to the Broker for RPC from those to the Broker for ACI. The trace levels are:

- 0 to switch off tracing.
- 1 to trace Broker calls.
- 2 to trace Broker calls and the payload.
- 3 to trace Broker calls and all buffers including the payload.

2 Redirect the trace to a file with the property `entirex.server.logfile`. Set this to the file name of the log file, the default is standard output.

Enabling Java Trace of SPM Plug-ins

In some cases a Java trace of the SPM plug-ins is needed to analyze an issue.

To enable Java trace of SPM plug-ins

1 Stop the Platform Manager. On UNIX it runs as a daemon.

2 Edit the file *custom_wrapper.conf in <Installation Dir>\profiles\SPM\configuration\custom_wrapper.conf*. Add the following line:

```
wrapper.java.additional.<n>=-Dentirex.trace=2
```

Example:

```
#encoding=UTF-8
# Configuration files must begin with a line specifying the encoding
# of the file.
# Put here your custom properties.
wrapper.successful_invocation_time=10
wrapper.java.initmemory=32
wrapper.restart.reload_configuration=TRUE
wrapper.java.additional.10=-Djava.util.Arrays.useLegacyMergeSort=true
wrapper.java.additional.20=-Dentirex.trace=2
```

In case of issues with SSL, add the line:

```
wrapper.java.additional.<n>=-Djavax.net.debug=ssl
```

Example with Java trace and SSL trace:

```
#encoding=UTF-8
# Configuration files must begin with a line specifying the encoding
# of the file.
# Put here your custom properties.
wrapper.successful_invocation_time=10
wrapper.java.initmemory=32
wrapper.restart.reload_configuration=TRUE
wrapper.java.additional.10=-Djava.util.Arrays.useLegacyMergeSort=true
wrapper.java.additional.20=-Dentirex.trace=2
wrapper.java.additional.30=-Djavax.net.debug=ssl
```

3 Restart the Platform Manager.
The Java trace is written to *<installation dir>\profiles\SPM\logs\wrapper.log*.

Tip: Search for string “EntireX Java Runtime” for the start of the trace.

To stop the Java trace of SPM plug-ins

1 Remove the additional lines in *<Installation Dir>\profiles\SPM\configuration\custom_wrapper.conf*.

2 Restart the Platform Manager.

15 EntireX Trace Utility

Introduction to the EntireX Trace Utility

Broker traces, as well as traces produced from applications communicating with the Broker (so-called "stub traces"), contain a lot of details of the particular Broker calls. However, their layout is different and not easy to understand. The EntireX Trace Utility reads these Broker kernel as well as stub traces and produces a file with a common layout, where one line corresponds to a Broker call. The file layout is a standard CSV file (comma-separated values).

The request (Broker call sent from the stub to the kernel) and the corresponding reply (response sent back from the kernel to the stub) are merged together and presented as one logical Broker call in one row of the output file. Line numbers in the trace file and times for the request and reply are provided. It is also possible to specify filters so only the specified subset of the Broker calls are extracted. Since the Broker trace file contains all activities from both clients and servers and since it is possible to filter the calls, an end-to-end analysis of a conversation is simple to analyze.

The EntireX Trace Utility is divided into two separate elements: Process Trace and Show Trace.

Process Trace

Process Trace is used to process the information contained in the Broker trace file, saving the requested output to a simple text file.

- Using the Tool
- Output Field Options
- Error Messages

Using the Tool

To open the EntireX Trace Utility under UNIX

- Run the script `traceutility.bsh` located under /<*Install_Dir*>/*EntireX*/*bin*.

To process the trace information

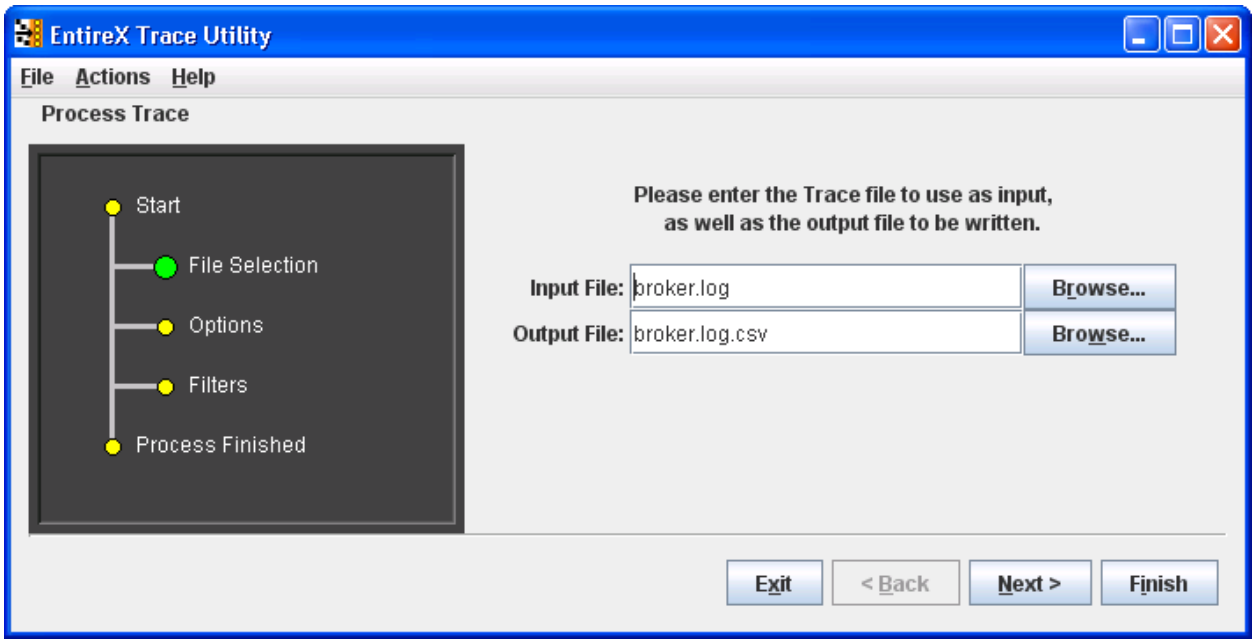
- Follow the instructions on the following screens:

  - File Selection
  - Options

- Filters

File Selection

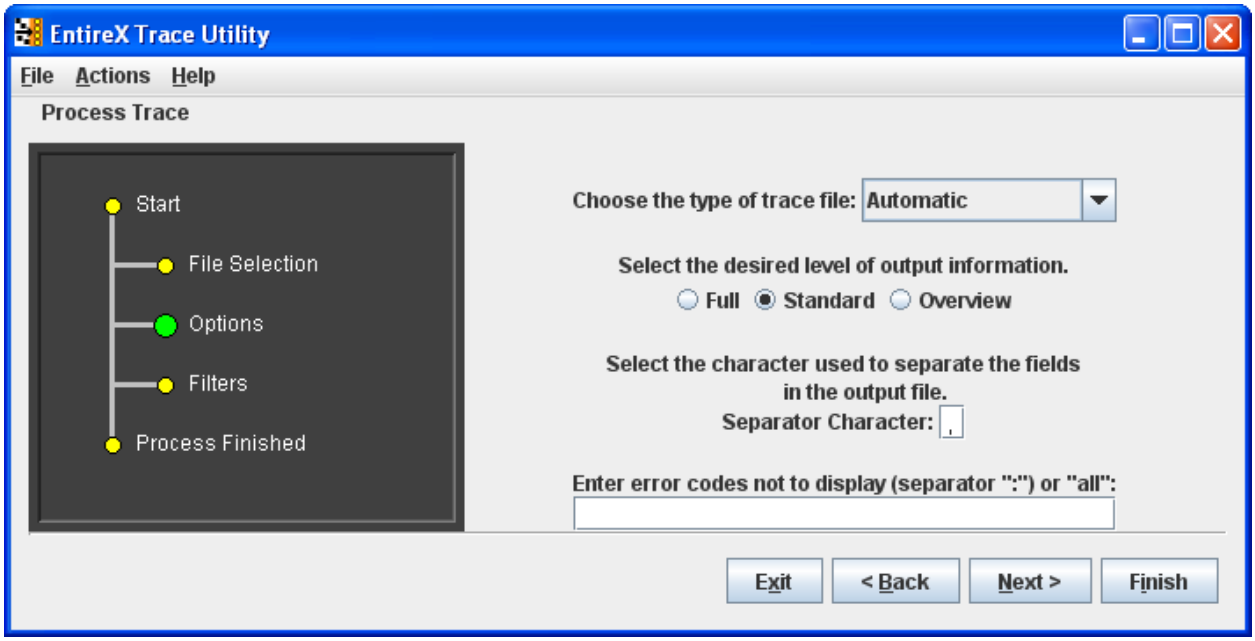
The following window is displayed.



The dark gray display section - the wizard window - shows you which step is required. **File Selection** has a large green dot, so the input and output files are required.

Options

In the display section, **Options** is green.



See *Output Field Options* for information on **Full**, **Standard** and **Overview**.

See *Options* under *Using the EntireX Trace Utility in Batch Mode* for information on type of trace file and error codes not to display.

The defaults of **Process Trace** are:

- use automatic detection of trace file type
- return the standard amount of output
- save the output fields separated with commas (as this format is needed to be able to view the output in Show Trace)
- display all errors found in the trace file.

The default separator character is ",", you can change this character.

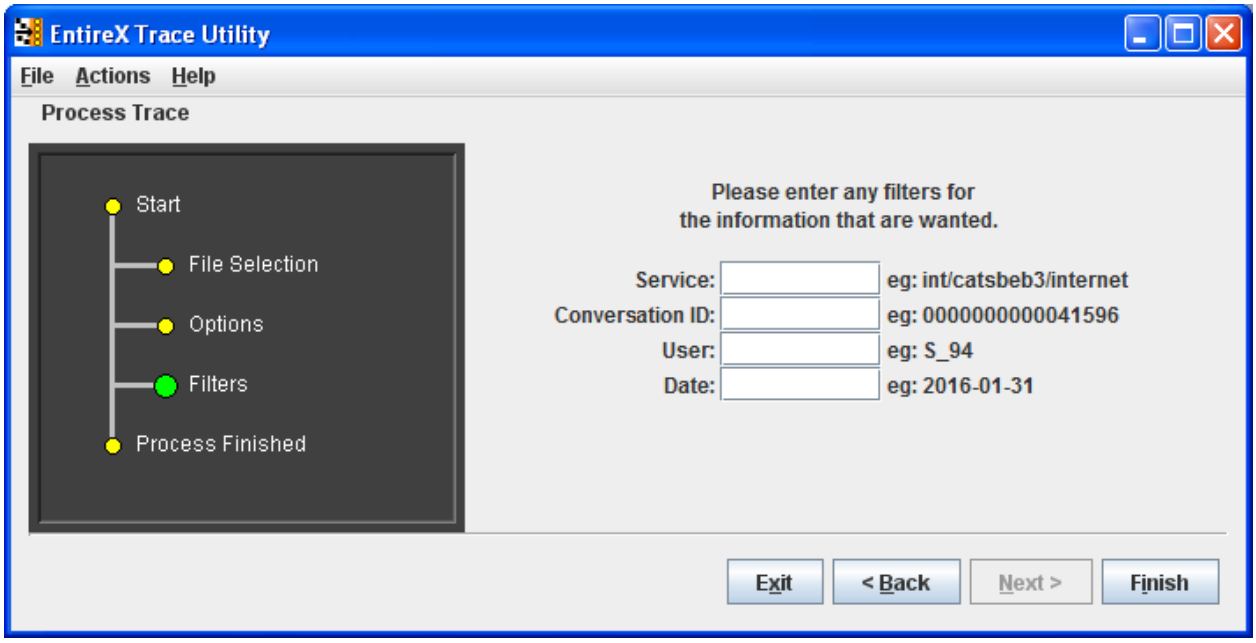
Filters

For the **Standard** and **Full** output options you can set filters to reduce the amount of information written to the output file.

You can set filters for the **Conversation ID** (for example: 000000000041596), the Broker **Service** (for example: int/catsbeb3/internet), the **User** (for example: S_94), and the **Date** for the call (for example 2016-01-31).

The **User** filter does not correspond to the User ID or Physical User ID from the trace, but a generated value from **Process Trace**. This filter can only be used after already analyzing an output file and deciding which User to filter for.

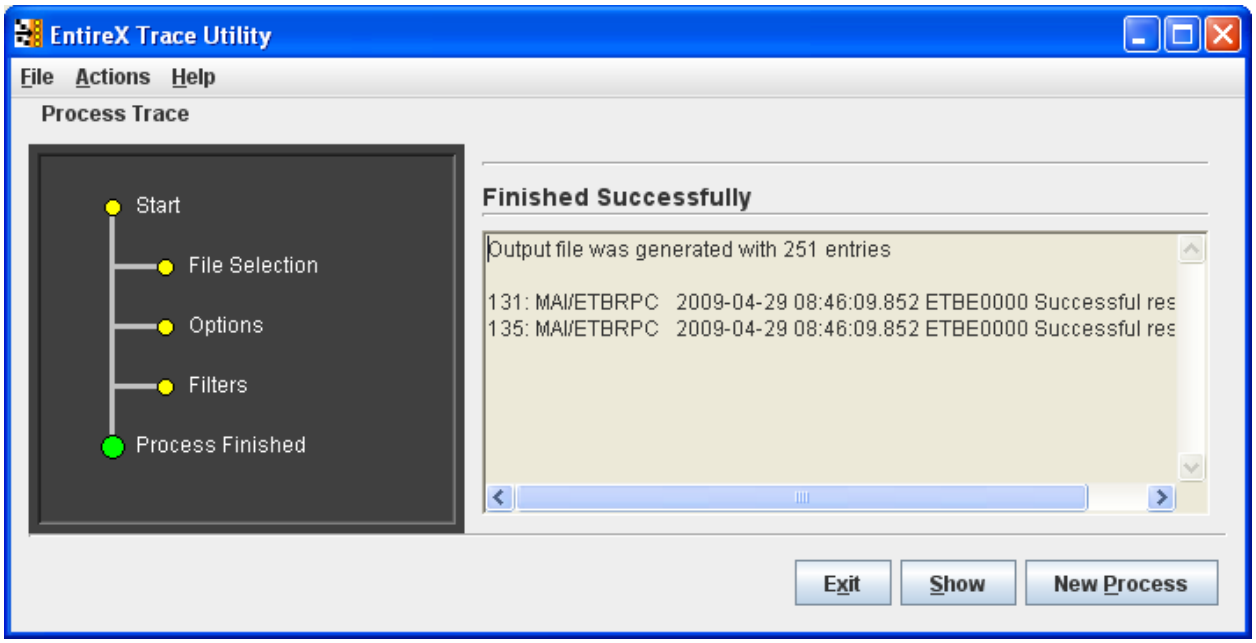
If more than one filter is specified, only those entries that satisfy all conditions will be displayed.



To generate the output file

- Choose **Finish**.

At this point any errors from processing the trace file are shown.



To display the results from the processing

- Choose **Show**.

To leave the program

- Choose **Exit**.

To process another trace file

- Choose **Process Trace** from the menu bar.

A new processing wizard is started.

Output Field Options

You may select between three levels of output to be written to the output file:

Option	Output Fields
Overview	Phys Userid, Userid, Token, User, Service
Standard	Thread, Req, Reply, Phys Userid, Userid, Token, User, Function, Error, Service, Convid, Uowid, Uowstatus, Slen, Retl, Cuid
Full	Thread, Req, Reply, Phys Userid, Userid, Token, User, Function, Error, Service, Convid, Uowid, Uowstatus, Slen, Retl, Cuid, Time1, Time2, Api, Rlen, Cstat, Charset, SecurityToken, Security, TimeDiff, ReplyTime, Seqid, AppName, Node, Stub, Library, Program, Brokerid, AppMon, Date, MessageIDRequest, CorrelationIDRequest, MessageIDReply, CorrelationIDReply, PartnerSeqid

Description of the columns in the CSV file (comma-separated values).

Note: Output which is the result of stub trace files does not contain entries for all columns.

Column	Explanation
Thread	The name of the Java thread executing the Broker call. Only available for trace files produced by the EntireX Java runtime.
Req	The line number in the trace file where the request part of the Broker call starts. 0 if the request cannot be found in the trace file.
Reply	The line number in the trace file where the reply part of the Broker call starts. 0 if the reply cannot be found in the trace file.
Phys.User ID	The physical user ID (Unique ID) which is displayed as a binary value in the Broker trace, nicely formatted. In case of a C stub trace file, the real physical user ID is not available; instead of this the thread ID is used to construct a replacement for the physical user ID.

Column	Explanation
User ID	The user ID of the Broker call.
Token	The token of the Broker call.
User	An artificial identifier for a user session (using physical user ID, user ID, and token). This is a unique number prefixed with either `C-` or `S-`. The latter will be used if the caller can be identified (using the available data in the trace) as a server application.
Function	The Broker function. If an option is specified it is appended to the function name. If a wait timeout is specified for the send or receive function it is appended.
Error	Error class, error number and error text. Error 0000 0000 is not displayed. The text "Successful response" is not displayed.
Service	The service address in the form class/server/service.
Convid	The conversation ID prefixed with *. If the conversation ID in the reply is different from the one in the request, the one from the reply is used.
Uowid	The unit of work ID prefixed with *. If the unit of work ID in the reply is different from the one in the request, the one from the reply is used.
Uowstatus	The unit of work status
Slen	The send length, i.e. the length of the data sent to the Broker.
Retl	The return length, i.e. the length of the data returned from the application.
Cuid	The client user ID (only for servers).
Time1	The time of the request entry in the trace file.
Time2	The time of the reply entry in the trace file.
Api	The API version.
Rlen	The (maximum) receive length specified in the send/receive call.
Cstat	The conversation status (only for servers).
Charset	The character set used by the caller. Typical values are `ascii`, `ebcdic siemens`. If a value for the locale string has been specified, it is added using / as a separator.
SecurityToken	An interpretation of the security token of the request part. If the reply also contains a security token it is added using / as a separator. The interpretation of the prefixes is as follows:

unknown	The security token cannot be identified as a security token valid for EntireX Security
enc	The send/receive data is encrypted.(1)
pwd	A password is specified in the call
newpwd	A new password is specified in the call.
stub	The security token has been built by an EntireX stub.
server	The security token has been processed by the Broker, the part which distinguishes security tokens is added.

Security	Some security-relevant control block fields of the call. If Forcelogon is enabled, "fl:" is displayed. If send/receive data is encrypted (SecurityToken, see above, is "enc") either "broker" or "target" is displayed. If a password has been specified an artificial

Column	Explanation
	password is displayed. If in addition a new password has been specified, it is added using / as a separator. The artificial password is displayed as "pwd" followed by a number (starting with 0).
TimeDiff	The elapsed time between the request and the reply (Time2 - Time1).
ReplyTime	Server response time (difference in time between the server receiving a request and sending the reply).
Seqid	The internal sequence ID of the Broker call. Only available for Broker version 7.3 or higher.
AppName	Name of the application communicating with the Broker. Only available if API version 9 or greater is used.
Node	Node name of the application which is communicating with the Broker, e.g. the TCP/IP hostname. Only available if API version 9 or greater is used.
Stub	Stub name and version used by the application communicating with the Broker. Only available if API version 9 or greater is used.
Library	Library name if Broker call is an RPC call. Only available for RPC clients, or for server version 8.0 or higher.
Program	Program name if Broker call is an RPC call. Only available for RPC clients, or for server version 8.0 or higher.
Brokerid	The Broker ID of the Broker call.
AppMon	Application Monitoring settings of the Broker call (for request and reply).
Date	The date of the request or reply entry in the trace file.
MessageIDRequest	The message ID of the request.
CorrelationIDRequest	The correlation ID of the request.
MessageIDReply	The message ID of the reply.
CorrelationIDReply	The correlation ID of the reply.
PartnerSeqid	The internal sequence ID of the related Broker call.

Notes:

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

Error Messages

The utility will only produce a meaningful result if the trace file is not corrupt. When transferring a trace from a mainframe, make sure all columns of the trace file are transferred, otherwise the utility might report errors (e.g. 2, 4 or 9). It is also possible that no errors are reported but the resulting CSV file has columns which contain invalid data.

Number	Message	Explanation
1	{0}	Text of a Java exception thrown at runtime.
2	Trace has incomplete entry for Binpart, expected length = {0}, actual length = {1}	Will be displayed a maximum of 5 times. Output for Security Token, Password, and New Password may be corrupted. Typical reason: columns in the trace file were lost when copying the trace from the mainframe.
3	Physical user ID {0} has wrong length	Trace file is corrupt.
4	Trace has incomplete entry for Key or Reply string	Will be displayed a maximum of 5 times. Output for any value may be corrupted. Typical reason: columns in the trace file were lost when copying the trace from the mainframe.
5	More then one request per user: {0}	This is an error condition similar to the Broker error 0037 0197.
6	does not include prefix	Trace file is corrupt.
7	does not include unique ID	Trace file is corrupt.
8	does not include reply or key	Trace file is corrupt.
9	Trace output might be incomplete and/or erroneous	Output for any value may be corrupt.
10	Problem with file {0}	Problem with trace or output file.
11	Not enough memory to process trace, try increasing -Xmx or split trace	The Java runtime does not have enough memory to process the trace file. Increase the memory or delete unnecessary sections in the trace file.
12	SeqID "{0}" does not match "{1}"	The sequence ID of the request and the reply do not match. This may happen if the trace file is incomplete or corrupted. Otherwise contact Software AG support and provide the trace file.
13	Found: {0}	The text of a Broker error message found in the trace file is displayed. All non-zero return codes and the result of `KERNELVERSION` calls are displayed. This can be configured using a tool parameter.

Show Trace

Show Trace enables you to display the values of a CSV file in a table (CSV=comma-separated values).

The first row of the file is used as the headers for the file.

Sorting the Information

The information in the tables can be sorted by descending or ascending order. The sorting is done alphabetically, not numerically.

To sort the information in a column by ascending order

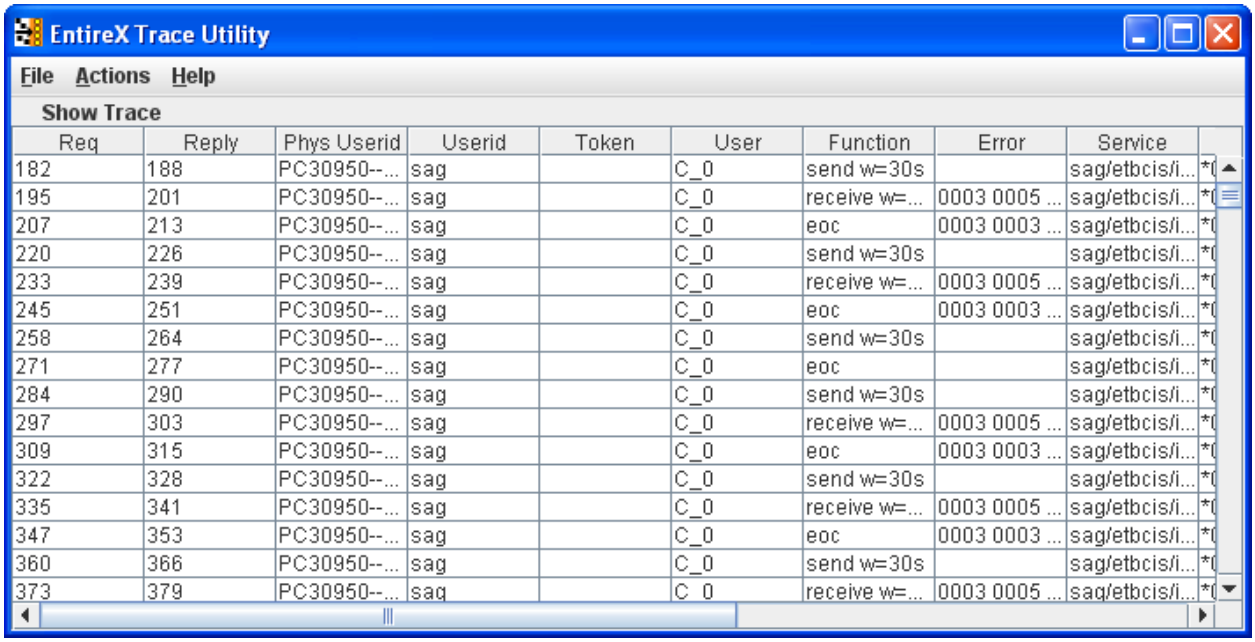
- Click on the header of the column.

To sort the information in a column by descending order

- Use SHIFT and click on the header of the column.

Loading and Saving a CSV File

You can load and save a CSV file using the options located in the File menu.



Req	Reply	Phys Userid	Userid	Token	User	Function	Error	Service
182	188	PC30950--...	sag		C_0	send w=30s		sag/etbcis/i...
195	201	PC30950--...	sag		C_0	receive w=...	0003 0005 ...	sag/etbcis/i...
207	213	PC30950--...	sag		C_0	eoc	0003 0003 ...	sag/etbcis/i...
220	226	PC30950--...	sag		C_0	send w=30s		sag/etbcis/i...
233	239	PC30950--...	sag		C_0	receive w=...	0003 0005 ...	sag/etbcis/i...
245	251	PC30950--...	sag		C_0	eoc	0003 0003 ...	sag/etbcis/i...
258	264	PC30950--...	sag		C_0	send w=30s		sag/etbcis/i...
271	277	PC30950--...	sag		C_0	eoc		sag/etbcis/i...
284	290	PC30950--...	sag		C_0	send w=30s		sag/etbcis/i...
297	303	PC30950--...	sag		C_0	receive w=...	0003 0005 ...	sag/etbcis/i...
309	315	PC30950--...	sag		C_0	eoc	0003 0003 ...	sag/etbcis/i...
322	328	PC30950--...	sag		C_0	send w=30s		sag/etbcis/i...
335	341	PC30950--...	sag		C_0	receive w=...	0003 0005 ...	sag/etbcis/i...
347	353	PC30950--...	sag		C_0	eoc	0003 0003 ...	sag/etbcis/i...
360	366	PC30950--...	sag		C_0	send w=30s		sag/etbcis/i...
373	379	PC30950--...	sag		C_0	receive w=...	0003 0005 ...	sag/etbcis/i...

Using the EntireX Trace Utility in Batch Mode

The EntireX Trace Utility is a graphical tool to process and display trace information. If the UNIX system does not have a graphical display (X-Windows), the EntireX Trace Utility can still be used as a command-line tool to process trace information.

To use the EntireX Trace Utility in batch mode

- Enter the following command in the command line:

```
java -jar exxutil.jar [-option] filename [
output file

]
```

or

```
java -Xms64m -Xmx256m -jar exxutil.jar [-option] filename [
output file

]
```

This specifies an initial and maximum memory allocation pool for the Java runtime (the defaults are 2 MB and 64 MB).

The *exxutil.jar* file is located in the classes subdirectory of the EntireX installation. *filename* is the name of the trace file. The output will be written to the file specified with the parameter *output file* or, if no name is specified there, output will be written to the file *filename.csv*.

Options

Option	Description	
-version	to display the version information	
-short	to generate an overview	
-long	to generate the full output	
-sep *char*	the separator character used in the resulting CSV file, default is ","	
-type *type*	By default the EntireX Trace Utility tries to infer the type of the trace file from the contents. If this is not possible (output shows "Processed 0 Broker calls") the type can be explicitly specified as follows:	
	java	The trace has been written by the EntireX Java runtime.
	cstub	The trace has been written by the C-based Broker stub.

Option	Description	
	`broker`	The trace has been written by the Broker kernel.
	`directrpc`	The trace has been written by the Direct RPC component of webMethods EntireX Adapter for Integration Server.
`-noshow` *param*	The utility displays all Broker errors found in the trace. To prevent this either all errors or a set of specified errors can be excluded from the display. To prevent the display of all errors specify "all" as parameter. To prevent the display of specific errors specify the 8 digit error class and number. Multiple errors can be specified separated by ":". Examples: `-noshow 00020002:00070007` or `-noshow "0074 0074"`.	

For the default and long display, filters can be specified:

Option	Description
`-user <` *user* `>`	to get entries for a particular user
`-conversation <` *convid* `>`	for a particular conversation ID
`-service`	for a particular service
`-date`	for a particular date

If more than one filter is specified, only those entries which satisfy all conditions will be displayed.

Example

```
java -jar exxutil.jar -long -sep ";" trace.txt
```

will generate all columns in a file trace.txt using ";" as separator character, the result will be in the file trace.txt.csv.

Usage Tips

Invalid or Incomplete Data in the Resulting CSV File

The utility will only produce a meaningful result if the trace file is not corrupt. When transferring a trace from a mainframe, make sure that all columns of the trace file are transferred. Otherwise the utility might report errors, e.g. error 2, 4 or 9. It may also happen that no errors are reported but the resulting CSV file has columns which contain invalid data.

Open the CSV File in Microsoft Excel

The CSV file can usually be opened in Microsoft Excel by double-clicking on the file name in the Windows Explorer. If the data is not displayed correctly, the separator character used by the utility (default is ",") does not match the list separator character used by Windows. Use the `-sep` option to specify a different separator character. To check the list separator used by Windows, go to **Control Panel > Regional Options > Numbers**.

Alternatively you may use the import functionality of Microsoft Excel. Open a spreadsheet, use **Data > Get External Data > Import Text File**. After selecting the file name (change default file type *.txt) the Text Import Wizard starts, which allows you to specify the delimiter (separator) character.

Displaying and Analyzing the CSV File in Microsoft Excel

The following are some tips how to use Microsoft Excel as a tool for displaying and analyzing the CSV file. They refer to Microsoft Excel 2000.

Formatting the spreadsheet: use CTRL A to select all data, change the font size e.g. to 8, then use **Format > Column > AutoFit Selection** to format all columns. Make the first line a "header line": select the 2nd line, use **Window > Freeze Panes**. Now, when scrolling through the entries the header line always stays on top.

Enable filtering: select the 1st line, use **Data > Filter > AutoFilter**. Now you have a drop-down box on each header entry that allows you to select a subset of the Broker calls.

Sorting Order

You can sort the entries in the generated CSV file using the Reply column. Thus the ordering corresponds to the time when the Broker kernel sends back the reply for the Broker call. Calls where no reply can be found in the trace appear at the end. If you use the Request column as the sorting criteria, the Broker calls will be ordered corresponding to the time when the Broker call arrives at the Broker kernel.

16 Broker Shutdown Statistics

Shutdown Statistics Output

After a successful Broker execution, shutdown statistics and related information are produced. This output is written in the following sequence:

1. The diagnostic message ETBD0444 is written into the Broker trace log.
2. The output - i.e. statistics, internals and user-specified parameters - is written into the end of the Broker trace log file at shutdown.

Table of Shutdown Statistics

See *Legend* below for explanation of output type.

Output Type	Display Field	Description
U	Broker ID	Identifies the Broker kernel to which the attribute file applies. See `BROKER-ID`.
I	Version	The version of the Broker kernel currently running.
I	Generated platform family	The platform family for which this Broker kernel was built.
I	Runtime platform	The platform on which this Broker kernel is currently running.
I	Start time	The date and time when this Broker kernel started.
S	Restart count	The restart count indicates how many times the Broker kernel has been started with the persistent store. Therefore, after a cold start `(PSTORE=COLD)`, the restart count will be 1. Then, after subsequent hot starts `(PSTORE=HOT)`, the restart count will be 2 or greater.
U	Trace level	The value for the trace setting for this Broker kernel. See `TRACE-LEVEL`.
U	Worker tasks	The number of worker tasks for this Broker kernel. See `NUM-WORKER`.
U	MAX-MEMORY	The value of `MAX-MEMORY` or 0 if not defined. See `MAX-MEMORY`.
S	Memory allocated	Size of the allocated memory, in bytes.
S	Memory allocated HWM	Highest size of allocated memory in bytes since Broker started.
U	NUM-SERVICE	Value of `NUM-SERVICE` or 0 if not defined. See `NUM-SERVICE`.
S	Services active	The number of services currently active for this Broker kernel.
U	NUM-CLIENT	Value of `NUM-CLIENT` or 0 if not defined. See `NUM-CLIENT`.
S	Clients active	The number of clients currently active for this Broker kernel.
S	Clients active HWM	The high watermark for the number of clients active for this Broker kernel.

Output Type	Display Field	Description
U	NUM-SERVER	Value of NUM-SERVER or 0 if not defined. See NUM-SERVER.
S	Servers active	The number of servers currently active for this Broker kernel.
S	Servers active HWM	The high watermark for the number of servers active for this Broker kernel.
U	NUM-CONVERSATION	Value of NUM-CONVERSATION or 0 if not defined. See NUM-CONVERSATION.
S	Conversations active	The number of conversations currently active for this Broker kernel.
S	Conversations active HWM	The high watermark for the number of conversations active for this Broker kernel.
U	NUM-LONG-BUFFER	Value of NUM-LONG-BUFFER or 0 if not defined. See NUM-LONG-BUFFER.
S	Long buffers active	The number of long message buffers currently in use for this Broker kernel.
S	Long buffers active HWM	The high watermark for the number of long message buffers used for this Broker kernel.
U	NUM-SHORT-BUFFER	Value of NUM-SHORT-BUFFER or 0 if not defined. See NUM-SHORT-BUFFER.
S	Short buffers active	The number of short message buffers currently in use for this Broker kernel.
S	Short buffers active HWM	The high watermark for the number of short message buffers used for this Broker kernel.
U	Persistent store type	The type of persistent store used by this Broker kernel. See PSTORE-TYPE.
U	UOW persistence	Indicates whether units of work are persistent or not in this Broker kernel. See STORE.
U	Persistent store startup	Indicates the status of the persistent store at Broker startup. See PSTORE.
U	Persistent status lifetime	The multiplier to compute the lifetime of the persistent status. See UWSTATP.
U	Deferred UOWs allowed	Indicates whether or not deferred units of work are allowed. See DEFERRED.
U	Maximum allowed UOWs	The maximum number of units of work that can be active concurrently for this Broker kernel. See MAX-UOWS.
U	Maximum messages per UOW	The maximum number of messages allowed in a unit of work. See MAX-MESSAGES-IN-UOW.
U	UOW lifetime in seconds	Indicates the default lifetime for a unit of work. See UOW-DATA-LIFETIME.
U	Maximum message length	Indicates the maximum message size that can be sent. See MAX-UOW-MESSAGE-LENGTH.

Output Type	Display Field	Description
U	New UOW messages allowed	Indicates whether or not new units of work are allowed in this Broker kernel. See NEW-UOW-MESSAGES.
S	UOWs active	The number of units of work currently active in this Broker kernel.
S	Current UOW	The number of the last unit of work in this Broker kernel.
U	Accounting	Indicates the status of accounting records in this Broker kernel. See ACCOUNTING.
U	SSL port *	If applicable, the SSL port number on which this Broker kernel will listen for connection requests. See SSL-specific attribute PORT.
U	TCP port *	If applicable, the TCP port number on which this Broker kernel will listen for connection requests. See TCP-specific attribute PORT.
I	Number of function calls	Marks the beginning of the section of summary statistics for all the function calls.
S	DEREGISTER	The number of Broker DEREGISTER function calls since startup.
S	EOC	The number of Broker EOC function calls since startup.
S	KERNELVERS	The number of Broker KERNELVERS function calls since startup.
S	LOGOFF	The number of Broker LOGOFF function calls since startup.
S	LOGON	The number of Broker LOGON function calls since startup.
S	RECEIVE	The number of Broker RECEIVE function calls since startup.
S	REGISTER	The number of Broker REGISTER function calls since startup.
S	SEND	The number of Broker SEND function calls since startup.
S	SYNCPOINT	The number of Broker SYNCPOINT function calls since startup.
S	UNDO	The number of Broker UNDO function calls since startup.
S	REPLY_ERROR	The number of Broker REPLY_ERROR function calls since startup.
I	Worker task statistics	Marks the beginning of the section of summary statistics for all the worker tasks.
I	Worker number	The identifier of the worker task.
I	Status	The status of the worker task at shutdown.
S	# of calls	The number of Broker calls handled by the worker task since startup.
S	Idle time in seconds	The number of seconds the worker task has been idle since startup.

* Does not apply to z/OS.

Legend

Output Type	Description	Value	Origin of Value
I	Internal Information	Static	Determined by Software AG EntireX.
S	Shutdown Statistic	Variable	Determined by Broker activity during execution.
U	User-Specified Parameter	Variable	Specified by Broker administrator before or, if allowable, during execution.

17 Command Logging in EntireX

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

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

Command Log Files

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

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

Under 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.

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

Filters need to be set before running the stub applications whose commands are to be logged.

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

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.

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

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.

```
etbcmd -blocalhost:1970:TCP -cDISABLE-CMDLOG-FILTER -dBROKER -xuser ↩
-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.

```
etbcmd -blocalhost:1970:TCP -cENABLE-CMDLOG-FILTER -dBROKER -xuser ↩
-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 file names `CMDLOG1` and `CMDLOG2` as the default command log file names.

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

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

To switch log files on demand, using etbcmd

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

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

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

18 Accounting in EntireX Broker

This chapter describes the accounting records for Broker that can be used for several purposes, including:

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

EntireX Accounting Data Fields

In the EntireX Accounting record, there are various types of data available for consumption by applications that process the accounting data:

Field Name	Accounting Version	Type of Field	Description
Record Write Time	1	A14 timestamp	The time this record was written to the accounting file in "YYYYMMDDHHMMSS" format.
EntireX Broker ID	1	A32	Broker ID from attribute file.
EntireX Version	1	A8	Version information, *v.r.s.p* where *v* =version *r* =release *s* =service pack *p* =patch level for example 10.5.0.00.
Platform of Operation	1	A32	Platform where EntireX is running.
EntireX Start Time	1	A14 timestamp	The time EntireX was initialized in "YYYYMMDDHHMMSS" format.
Accounting Record Type	1	A1	It is always C for conversation. Future Types will have a different value in this field.
Client User ID	1	A32	`USER-ID` ACI field from the client in the conversation.
Client Token	1	A32	`TOKEN` field from the ACI from the client.
Client Physical ID	1	A56	The physical user ID of the client, set by EntireX.
Client Communication Type	1	I1	Communication used by client: 1 = Net-Work

Field Name	Accounting Version	Type of Field	Description
			2 = TCP/IP 3 = APPC 4 = IBM® MQ 5 = SSL
Client Requests Made	1	I4	Number of Requests made by client.
Client Sent Bytes	1	I4	Number of bytes sent by client.
Client Received Bytes	1	I4	Number of bytes received by client.
Client Sent Messages	1	I4	Number of messages sent by client.
Client Received Messages	1	I4	Number of messages received by client.
Client Sent UOWs	1	I4	Number of UOWs sent by client.
Client UOWs Received	1	I4	Number of UOWs received by client.
Client Completion Code	1	I4	Completion code client received when conversation ended.
Server User ID	1	A32	`USER-ID` ACI field from the server in the conversation.
Server Token	1	A32	`TOKEN` field from the ACI from the server.
Server Physical ID	1	A56	The physical user ID of the server, set by EntireX.
Server Communication Type	1	I1	Communication used by Server: 1 = Entire Net-Work 2 = TCP/IP 3 = APPC 4 = IBM® MQ 5 = SSL
Server Requests Made	1	I4	Number of requests made by server.
Server Sent Bytes	1	I4	Number of bytes sent by server.
Server Received Bytes	1	I4	Number of bytes received by server.
Server Sent Messages	1	I4	Number of messages sent by server.
Server Received Messages	1	I4	Number of messages received by server.
Server Sent UOWs	1	I4	Number of UOWs sent by server.
Server Received UOWs	1	I4	Number of UOWs received by server.
Server Completion Code	1	I4	Completion code server received when conversation ended.
Conversation ID	1	A16	`CONV-ID` from ACI.
Server Class	1	A32	`SERVER-CLASS` from ACI.
Server Name	1	A32	`SERVER-NAME` from ACI.
Service Name	1	A32	`SERVICE` from ACI.
CID=NONE Indicator	1	A1	Will be N if `CONV-ID=NONE` is indicated in application.

Field Name	Accounting Version	Type of Field	Description
Restarted Indicator	1	A1	Will be R if a conversation was restarted after a Broker shutdown.
Conversation Start Time	1	A14 timestamp	The time the conversation began in "YYYYMMDDHHMMSS" format.
Conversation End Time	1	A14 timestamp	The time the conversation was cleaned up in "YYYYMMDDHHMMSS" format.
Conversation CPU Time	1	I4	Number of microseconds of CPU time used by the conversation
Client Security Identity	2	A32	Actual identity of client derived from authenticated user ID.
Client Application Node	2	A32	Node name of machine where client application executes.
Client Application Type	2	A8	Stub type used by client application.
Client Application Name	2	A64	Name of the executable that called the broker. Corresponds to the Broker Information Service field `APPLICATION-NAME`.
Client Credentials Type	2	I1	Mechanism by which authentication is performed for client.
Server Security Identity	2	A32	Actual identity of server derived from authenticated user ID.
Server Application Node	2	A32	Node name of machine where server application executes.
Server Application Type	2	A8	Stub type used by server application.
Server Application Name	2	A64	Name of the executable that called the broker. Corresponds to the Broker Information Service field `APPLICATION-NAME`.
Server Credentials Type	2	I1	Mechanism by which authentication is performed for server.
Client RPC Library	3	A128	RPC library referenced by client when sending the only/first request message of the conversation.
Client RPC Program	3	A128	RPC Program referenced by client when sending the only/first request message of the conversation.
Server RPC Library	3	A128	RPC library referenced by server when sending the only/first response message of the conversation.
Server RPC Program	3	A128	RPC Program referenced by server when sending the only/first response message of the conversation.
Client IPv4 Address	4	A16	IPv4 address of the client.
Server IPv4 Address	4	A16	IPv4 address of the server.

Field Name	Accounting Version	Type of Field	Description
Client Application Version	4	A16	Application version of the client.
Server Application Version	4	A16	Application version of the server.
Client IPv6 Address	5	A46	IPv6 address of the client.
Server IPv6 Address	5	A46	IPv6 address of the server.

Note: Accounting fields of any version greater than 1 are created only if the attribute `ACCOUNTING-VERSION` value is greater than or equal to the corresponding version. For example: accounting fields of version 2 are visible only if `ACCOUNTING-VERSION=2` or higher is specified.

Using Accounting under UNIX and Windows

Broker Attribute File Settings

ACCOUNTING = NO | YES | (YES, SEPARATOR=Separator Characters) (Default is NO)

Set this parameter to "NO" (i.e., do not create accounting data) or "YES" to create accounting data. Up to seven separator characters can specified using the SEPARATOR suboption, for example ACCOUNTING = (YES, SEPARATOR=;). If no separator character is specified, the comma character will be used.

Retrieving Accounting Data

The accounting file will be located in the Broker's installed directory. The file's name is based on the ETB_LOG environment variable and the current date and time (for uniqueness). Example: If ETB_LOG is set to BROKER1.LOG, the accounting data file will be named BROKER1_*YYYYMMDDHHMMSS*.csv. If ETB_LOG is not set, the Broker's ID will be used, with an extension of CSV (e.g. ETB048_*YYYYMMDDHHMMSS*.csv). See *Environment Variables in EntireX*.

Example Uses of Accounting Data

Chargeback

Customers can use the EntireX accounting data to perform chargeback calculations for resource utilization in a data center. Suppose EntireX Broker is being used to dispatch messages for three business departments: Accounts Receivable, Accounts Payable, and Inventory. At the end of each month, the customer needs to determine how much of the operation and maintenance cost of EntireX Broker should be assigned to these departments. For a typical month, assume the following is true:

Department	Amount of Data	Percentage	Messages Sent	Percentage	Average Percentage
Accts Payable	50 MB	25	4000	20	22.5
Accts Receivable	40 MB	20	6000	30	25
Inventory	110 MB	55	10000	50	52.5

The use of Broker resources here is based upon both the amount of traffic sent to the Broker (bytes) as well as how often the Broker is called (messages). The average of the two percentages is used to internally bill the departments, so 52.5% of the cost of running EntireX Broker would be paid by the Inventory Department, 25% by the Accounts Receivable Department, and 22.5% by the Accounts Payable Department.

Trend Analysis

The Accounting Data can also be used for trend analysis. Suppose a customer has several point-of-sale systems in several stores throughout the United States that are tied into the corporate inventory database with EntireX. The stubs would be running at the stores, and the sales data would be transmitted to the Broker, which would hand it off to the appropriate departments in inventory. If these departments wish to ascertain when the stores are busiest, they can use the accounting data to monitor store transactions. Assume all of the stores are open every day from 9 AM to 10 PM.

Local Time	Average: Weekday Transactions per Store	Maximum Weekday Transactions in any Store	Average Weekend Transactions per Store	Maximum Weekend Transactions in any Store
9 AM	7.3	27	28.2	83
10 AM	11.2	31	29.3	102
11 AM	14.6	48	37.9	113
12 noon	56.2	106	34.8	98
1 PM	25.6	65	34.2	95
2 PM	17.2	52	38.5	102
3 PM	12.1	23	42.7	99
4 PM	18.3	34	43.2	88
5 PM	26.2	47	45.2	93
6 PM	38.2	87	40.6	105
7 PM	29.6	83	39.2	110
8 PM	18.6	78	28.6	85
9 PM	11.2	55	17.5	62

The owner of the stores can examine the data and make decisions based upon the data here. For example, on weekdays, he or she can see that there is little business until lunchtime, when the number of transactions increase. It then decreases during lunch hour; then there is another increase from 5 PM to 8 PM, after people leave work. Based on this data, the owner might investigate changing the store hours on weekdays to 10 AM to 9 PM. On the weekend the trends are different, and the store hours could be adjusted as well, although there is a more regular customer flow each hour on the weekends.

Tuning for Application Performance

Assume that a customer has two applications that perform basic request/response messaging for similar sized messages. The applications consist of many Windows PC clients and Natural RPC Servers on UNIX. An analysis of the accounting data shows the following:

Application Type	Class	Server	Service	Average Server Messages Received per Conversation	Average Client Messages Received per Conversation
Application 1:	CLASS1	SERVER1	SERVICE1	10.30	10.29
Application 2:	CLASS2	SERVER2	SERVICE2	10.30	8.98

A further analysis of the accounting data reveals that there are a lot of non-zero response codes in the records pertaining to Application 2, and that a lot of these non-zero responses indicate timeouts. With that information, the customer can address the problem by modifying the server code, or by adjusting the timeout parameters for SERVER2 so that it can have more time to get a response from the Service.