webMethods EntireX

Administration under BS2000

Version 10.9

April 2023

WEBMETHODS

This document applies to webMethods EntireX Version 10.9 and all subsequent releases.

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

Copyright © 1997-2023 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

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

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

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

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

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

Document ID: EXX-ADMIN_BS2000-109-20230403

Table of Contents

EntireX Administration under BS2000 v
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
2 Setting up Broker Instances 5
Setting up TCP/IP Transport 6
Setting up Entire Net-Work/Adabas SVC Transport 6
Starting and Stopping the Broker 6
Tracing EntireX Broker 9
Protecting a Broker against Denial-of-Service Attacks 11
3 Broker Attributes 13
Name and Location of Attribute File 15
Attribute Syntax 15
Broker-specific Attributes 17
Service-specific Attributes 34
Codepage-specific Attributes 44
Adabas SVC/Entire Net-Work-specific Attributes 46
Security-specific Attributes 49
TCP/IP-specific Attributes 51
SSL/TLS-specific Attributes 54
Adabas-specific Attributes 59
Application Monitoring-specific Attributes 61
Variable Definition File 62
4 Configuring Broker for Internationalization 63
Configuring ICU Conversion 64
Building and Installing ICU Custom Converters 65
Writing Translation User Exits 65
Configuring Translation User Exits 67
Configuring Translation 68
5 Managing the Broker Persistent Store 69
Implementing an Adabas Database as Persistent Store 70
6 Broker Resource Allocation 77
General Considerations 78
Specifying Global Resources 78
Restricting the Resources of Particular Services 79
Specifying Attributes for Privileged Services 81
Maximum Units of Work 81
Calculating Resources Automatically 82
Dynamic Memory Management 84
Dynamic Worker Management 85
Storage Report 87
Maximum TCP/IP Connections per Communicator 88

7 EntireX Broker Security Server for BS2000 89
Activating Authentication 90
Starting the Broker Security Server 90
Stopping the Broker Security Server 91
Tracing with the Broker Security Server 91
Broker Security Server Parameters 92
8 Administering Broker Stubs 93
Available Stub 94
Linking the Stubs 94
Transport Methods for Broker Stubs 97
Using Job Variables 100
Using BROKER under openUTM 100
9 Broker Command-line Utilities 101
ETBINFO 102
ETBCMD 108
10 Operator Commands 115
Command Syntax 116
General Broker Commands 116
Participant-specific Commands 121
Security-specific Commands 126
Transport-specific Commands 127
XCOM-specific Commands 131
11 Administering the RPC Server for BS2000 135
Customizing the RPC Server 136
Configuring the RPC Server 137
Locating and Calling the Target Server 144
Starting the RPC Server 145
Stopping the RPC Server 145
Activating Tracing for the RPC Server 153
12 Tracing EntireX Components under BS2000 149
Tracing EntireX Broker 150
Tracing Broker Stubs 151
Activating Tracing for the RPC Server 153
Tracing Broker Security Server 153
13 Broker Shutdown Statistics 155
Shutdown Statistics Output 156
Table of Shutdown Statistics 156
14 Command Logging in EntireX 161
Introduction to Command Logging 162
ACI-driven Command Logging 164
Dual Command Log Files 164
15 Accounting in EntireX Broker 165
EntireX Accounting Data Fields 166
Example Uses of Accounting Data 169

EntireX Administration under BS2000

Broker Configuration	Broker-related configuration topics.
Broker Add-ons	Broker stubs, command-line utilities, operator commands.
Administration	Administering the RPC Server for BS2000.
Logging and Tracing EntireX	Logging, tracing and accounting.

See also *Introduction to EntireX Mainframe Broker Monitoring*.

1 About this Documentation

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

Document Conventions

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

Online Information and Support

Product Documentation

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

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

Product Training

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

Tech Community

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

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

Product Support

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

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

Data Protection

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

2 Setting up Broker Instances

- Setting up TCP/IP Transport 6
- Setting up Entire Net-Work/Adabas SVC Transport 6
- Starting and Stopping the Broker 6
- Tracing EntireX Broker 9
- Protecting a Broker against Denial-of-Service Attacks 11

This chapter contains information on setting up the Broker under BS2000. It assumes that you have completed the relevant steps described under *Installing EntireX under BS2000*.

Setting up TCP/IP Transport

The recommended way to set up the TCP/IP communicator is to define `PORT=nnnn` and optionally `HOST=x.x.x.x|hostname` under *TCP/IP-specific Attributes*.

However, if no port number is specified in the Broker attribute file, the broker kernel will default port number of 1971. This is the same default port number that the stubs use.

Setting up Entire Net-Work/Adabas SVC Transport

To set up EntireX Net-Work communication mechanism

1. Ensure that appropriate values are supplied in the broker attribute file section `DEFAULTS=NET`, paying particular attention to the `IUBL` parameter - which specifies the maximum send/receive buffer length that can be sent between an application and Broker kernel within a single request - and `NABS`, which governs the total amount of storage available concurrently for all users communicating over this transport mechanism. See *Adabas SVC/Entire Net-Work-specific Attributes*.
2. Ensure that communication with the broker is possible by running the installation verification programs (bcoc, bcos) using transport type NET. See *Verifying the Installation of the Broker* in the BS2000 Installation documentation.

Starting and Stopping the Broker

Starting the Broker

To start the broker

- Enter the following SDF command:

```
/ENTER-PROCEDURE *LIB(LIB=EXX103.JOBS,ELE=START-BROKER), -
/JOB-NAME=ETB,LOGGING=*NO,RESOURCES=*PAR(CPU-LIMIT=*NO)
```

We recommend using a three-character job name. The job name is taken as prefix for all subsequently started tasks. Because the job name is limited to eight characters, a longer job name will overwrite the suffix added by EntireX Broker. For example: EntireX Broker running with three worker tasks and NET-TCP communication, `JOB-NAME=ETB`, `CPU-LIMIT=*NO`:

```
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
ETB      5397 2 BATCH    9 255      2.2379  NTL    1
ETBCOM   5398 2 BATCH    9 255      1.3577  NTL    1
ETBWRK00 5399 2 BATCH    9 255      0.8970  NTL    1
ETBWRK01 5400 2 BATCH    9 255      0.7571  NTL    1
ETBWRK02 5401 2 BATCH    9 255      0.7445  NTL    1
ETBTCP00 5402 2 BATCH    9 255      0.6124  NTL    1
ETBTCPPX 5403 2 BATCH    9 255      0.5417  NTL    1
ETBNET00 5404 2 BATCH    9 255      0.6555  NTL    1
ETBTOM   5407 2 BATCH    9 255      6.4044  NTL    1
```

The properties assigned to the main task (ETB), e.g. `JOB-CLASS`, `CPU-LIMIT`, will be inherited by all subsequently started tasks. For `CPU-LIMIT`, if specified, only `*NO` (no time limit) and `*STD` are inherited.

Stopping the Broker

To stop the broker from a privileged user ID

- Enter the following command:

```
/INFORM-PROGRAM MSG='ETBSTOP',JOB-IDENTIFICATION=*TSN(TSN=tsn)
```

where *tsn* is the task number associated with the broker main task (in the example above the TSN of job name ETB)

All other tasks that were created as a result of starting the broker will be stopped automatically.

To stop the broker from an operator console

- Enter the following command:

```
/INTR tsn,ETBSTOP
```

where *tsn* is the task number associated with the broker main task (in the example above the TSN of job name ETB)

All other tasks that were created as a result of starting the broker will be stopped automatically.

To stop the broker from a non-privileged user ID

- Use the S-procedure `STOP-BROKER` in EXX103.JOBS

Startup Parameter	Description	Default

`BROKER-ID`	Depending on the communication method, the Broker ID can be specified in two different formats:
	■ **TCP Transport Method**
	`ip:port:TCP`
	where *ip* is the address or DNS host name,
	port is the port number that EntireX Broker is listening on, and
	`TCP` is the protocol name
	■ **NET Transport Method**
	`ETBnnn:SVCmmm:NET`
	where *nnn* is the ID under which EntireX Broker is connected to the Adabas ID table,
	mmm is the SVC number under which the Adabas ID table can be accessed, and
	`NET` is the protocol name
		none

`ADABAS-PARAMETERS`	Adabas parameters used for NET communication method.	ETB-ADAPARM
`USERID`	If EntireX Broker is running with EntireX Security, a user ID needs to be supplied.	none
PASSWORD	If EntireX Broker is running with EntireX Security, a password needs to be supplied.	none
EXX-LIB	EntireX Broker module library.	EXX103.LIB
EXX-JOBS	EntireX Broker jobs library.	EXX103.JOBS
WAL-MOD	WAL module library.	WAL842.MOD

Set the broker ID in the PARAMETER-DECLARATION section and enter following command:

```
/CALL-PROCEDURE (EXX103.JOBS, STOP-BROKER)
```

Tracing EntireX Broker

This section covers the following topics:

- Broker TRACE-LEVEL Attribute
- Attribute File Trace Setting
- Deferred Tracing
- Dynamically Switching On or Off the EntireX Broker Trace

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	
Conversion trace level	DEFAULTS=SERVICE; Trace option of the service-specific broker attribute CONVERSION.	
Security trace level	DEFAULTS=SECURITY	1
Transport trace level	DEFAULTS=NET | TCP	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 operator command TRACE.

Trace messages are written to the SYSOUT file of the EntireX Broker common output manager (COM) task.

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`.
- **Operator Command**
  Issue an operator command. See `TRACE`.

  See also *Deferred Tracing*.

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 Broker Attributes

- Name and Location of Attribute File 15
- Attribute Syntax 15
- Broker-specific Attributes 17
- Service-specific Attributes 34
- Codepage-specific Attributes 44
- Adabas SVC/Entire Net-Work-specific Attributes 46
- Security-specific Attributes 49
- TCP/IP-specific Attributes 51
- SSL/TLS-specific Attributes 54
- Adabas-specific Attributes 59
- Application Monitoring-specific Attributes 61
- Variable Definition File 62

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

Note: This section lists EntireX Broker parameters applicable to operating system BS2000 only. For attributes applicable to *all* operating systems, see *Broker Attributes* in the platform-independent Administration documentation.

Name and Location of Attribute File

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

Platform	File Name/Location
BS2000	File *ETB-ATTR* in library *EXX103.JOBS*.

Attribute Syntax

Each entry in the attribute file has the format:

```
ATTRIBUTE-NAME=value
```

The following rules and restrictions apply:

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

- If a variable has no value: if the variable name is enclosed in braces, error 00210594 is given, otherwise `$variable name` will be used as the variable value.
- If you encounter problems with braces (and this is quite possible in a z/OS environment), we suggest you omit the braces.

Broker-specific Attributes

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

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Determines whether accounting records are created. 1 Collect accounting information. This value is supported for reasons of compatibility with EntireX Broker 7.2.1 and below. 2 Collect extended accounting information in addition to that available with option 1. 3 Create accounting records in layout of version 3. 4 Create accounting records in layout of version 4. 5 Create accounting records in layout of version 5. This parameter applies when ACCOUNTING is activated.					
ACI-CONVERSION	YES | NO	O	z	u	w	b
	Determines the handling of ACI request and response strings of USTATUS. YES Convert ACI request and response strings with ICU. See *ICU Conversion* in the Internationalization documentation. NO Translate ACI request and response with internal translation table without support of national characters. See *Translation User Exit* in the Internationalization documentation. **Note:** This attribute was undocumented in EntireX versions prior to 10.3 and had default value NO. This meant that a translation user exit was used instead; this is no longer recommended.					
APPLICATION-MONITORING or APPMON	YES | NO	O	z	u	w	b
	Enable application monitoring in EntireX Broker. YES Enable application monitoring. NO Disable application monitoring. See the separate Application Monitoring documentation.					
AUTOLOGON	YES | NO	O	z	u	w	b
	YES LOGON occurs automatically during the first SEND or REGISTER. NO The application has to issue a LOGON call.					
BLACKLIST-PENALTY-TIME	5M | *n* | *n*S | *n*M | *n*H	R	z	u	w	b
	Define the length of time a participant is placed on the PARTICIPANT-BLACKLIST to prevent a denial-of-service attack. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647). *n*M Non-activity time in minutes (max. 35791394).					

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	*n*M Interval in minutes (max. 35791394). *n*H Interval in hours (max. 596523). The minimum value is 16 seconds. We strongly recommend the default value (60 seconds), except for very slow machines.					
CONV-DEFAULT	UNLIM | *n*	O	z	u	w	b
	Default number of conversations that are allocated for every service. UNLIM The number of conversations is restricted only by the number of conversations globally available. Precludes the use of NUM-CONVERSATION. *n* Number of conversations. This value can be overridden by specifying a CONV-LIMIT for the service. A value of 0 (zero) is invalid.					
DEFERRED	NO | YES	O	z	u	w	b
	Disable or enable deferred processing of units of work. NO Units of work cannot be sent to the service until it is available. YES Units of work can be sent to a service that is not up and registered. They will be processed when the service becomes available.					
DYNAMIC-MEMORY-MANAGEMENT	YES | NO	O	z	u	w	b
	YES An initial portion of memory is allocated at broker startup based on defined NUM-* attributes or internal default values if no NUM-* attributes have been defined. More memory is allocated without broker restart if there is a need to use more storage. Unused memory is deallocated. The upper limit of memory consumption can be defined by the attribute MAX-MEMORY. See *Dynamic Memory Management* under *Broker Resource Allocation* in the platform-independent Administration documentation. NO All memory is allocated at broker startup based on the calculation from the defined NUM-* attributes. Size of memory cannot be changed. This was the known behavior of EntireX 7.3 and earlier. If you run your broker with attribute DYNAMIC-MEMORY-MANAGEMENT=YES, the following attributes are not needed: ■ CONV-DEFAULT ■ NUM-CONV[ERSATION] ■ HEAP-SIZE ■ NUM-LONG[-BUFFER] ■ LONG-BUFFER-DEFAULT ■ NUM-SERVER ■ SERVER-DEFAULT ■ NUM-SERVICE-EXTENSION ■ SHORT-BUFFER-DEFAULT ■ NUM-SERVICE					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	■ NUM-CLIENT ■ NUM-SHORT[-BUFFER] ■ NUM-CMDLOG-FILTER ■ NUM-UOW|MAX-UOWS|MUOW ■ NUM-COMBUF ■ NUM-WQE **Caution:** However, if one of these attributes is defined, it determines the allocation size of that particular broker resource.					
DYNAMIC-WORKER-MANAGEMENT	NO | YES	O	z	u	w	b
	NO All worker tasks are started at broker startup. The number of worker tasks is defined by NUM-WORKER. After this initial step, no further worker tasks can be started. This is default and simulates the behavior of EntireX version 8.0 and earlier. YES As above, the initial portion of worker tasks started at broker startup is determined by NUM-WORKER. However, if there is a need to handle an increased workload, additional worker tasks can be started at runtime without restarting broker. Conversely, if a worker task remains unused, it is stopped. The upper and lower limit of running worker tasks can be defined by the attributes WORKER-MIN and WORKER-MAX. If you run broker with DYNAMIC-WORKER-MANAGEMENT=YES, the following attributes are useful to optimize the overall processing: ■ WORKER-MAX ■ WORKER-MIN ■ WORKER-NONACT ■ WORKER-QUEUE-DEPTH ■ WORKER-START-DELAY The attribute NUM-WORKER defines the initial number of worker tasks started during initialization. See *Dynamic Worker Management*.					
ETBCOM	NO | YES	O	z	u	w	
	YES | NO	O				b
	Bundles the output of the various broker tasks in task ETBCOM.					
HEAP-SIZE	1024 | *n*	O	z	u	w	b
	Defines the size of the internal heap in KB. Not required if you are using DYNAMIC-MEMORY-MANAGEMENT. If you are *not* using dynamic memory management, we strongly recommend specifying - as a minimum - the default value of 1024 KB.					
ICU-CONVERSION	YES | NO	O	z	u	w	b
	Disable or enable ICU conversion. YES ICU is loaded and available for conversion. It is a prerequisite for CONVERSION=SAGTCHA and CONVERSION=SAGTRPC.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	NO ICU is not loaded and not available for conversion. CONVERSION=SAGTCHA and CONVERSION=SAGTRPC cannot be used. If any of the broker service definitions uses the character conversion approach *ICU Conversion*, that is, CONVERSION=SAGTCHA or CONVERSION=SAGTRPC, ICU-CONVERSION must be set to YES. If you are using only a user exit (see *User Exits* under *Introduction* in the Internationalization documentation) or CONVERSION=NO as character conversion approach for all your broker service definitions, ICU-CONVERSION can be set to NO. ICU requires additional storage to run properly. If ICU conversion is not needed, setting ICU-CONVERSION to NO will help to avoid unnecessary storage consumption.					
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	b
	Number of long buffers to be allocated for each service. UNLIM The number of long message buffers is restricted only by the number of buffers globally available. Precludes the use of NUM-LONG-BUFFER. *n* Number of buffers. This value can be overridden by specifying a LONG-BUFFER-LIMIT for the service. A value of 0 (zero) is invalid.					
MAX-MEMORY	0 | *n* | *n*K | *n*M | *n*G | UNLIM	O	z	u	w	b
	Defines the upper limit of memory allocated by broker if DYNAMIC-MEMORY-MANAGEMENT=YES has been defined. 0, UNLIM No memory limit. others Defines the maximum limit of allocated memory. If limit is exceeded, error 671 “Requested allocation exceeds MAX-MEMORY” is generated.					
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	b
	Maximum message size that the broker kernel can process. This value is transport-dependent. The default value represents the highest positive number that can be stored in a four-byte integer.					
MAX-MESSAGES-IN-UOW	16 | *n*	O	z	u	w	b
	Maximum number of messages in a unit of work.					
MAX-MSG	See MAX-MESSAGE-LENGTH.					

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

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	`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	b
	Defines the number of short message containers. Short message containers have a fixed length of 256 bytes and are used to store requests of no more than 2048 bytes. To store a request of 1024 bytes, for example, would require four short message containers. *n* Number of buffers. `AUTO` Uses the `SHORT-BUFFER-DEFAULT` and the service-specific `SHORT-BUFFER-LIMIT` values to calculate the number of short message buffers. Do not set the values used in the calculation to `UNLIM`. **Note:** 1. In *non-conversational* mode, message containers are released as soon as the client receives a reply from the server. If no reply is requested, message containers are released as soon as the server receives the client request. 2. In *conversational* mode, the last message received is always kept until a new one is received. 3. If a wildcard service is defined in the service-specific section of the attribute file, the value of `AUTO` is invalid. 4. See ***Wildcard Service Definitions***.					
`NUM-UOW`	`0` | *n*	O	z	u	w	b
	The maximum number of UOWs that can be concurrently active broker-wide. The default value is 0 (zero), which means that the broker will process only messages that					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	are not part of a unit of work. If UOW processing is to be done by any service, a NUM-UOW value must be 1 or larger for the broker. (MAX-UOWS is an alias for this attribute.) The NUM-UOW value for the service will default to the value set for the broker.					
NUM-WORKER	1 | *n* (max. 64)	R	z	u	w	b
	Number of worker tasks that the broker can use. The number of worker tasks determines the number of functions (SEND, RECEIVE, REGISTER, etc.) that can be processed concurrently. At least one worker task is required; this is the default value.					
NUM-WQE	1-32768	R	z	u	w	b
	Maximum number of requests that can be processed by the broker in parallel, over all transport mechanisms. Each broker command is assigned a worker queue element, regardless of the transport mechanism being used. This element is released when the user has received the results of the command, including the case where the command has timed out.					
PARTICIPANT-BLACKLIST	YES | NO	R	z	u	w	b
	Determines whether participants attempting a denial-of-service attack on the broker are to be put on a blacklist. YES Create a participant blacklist. NO Do not create a participant blacklist. See *Protecting a Broker against Denial-of-Service Attacks* in the platform-specific Administration documentation.					
PERCENTAGE-FOR-CONNECTION-SHORTAGE-MESSAGE	90 | 1-100	O	z	u	w	b
	Broker will issue a message if the defined percentage value of TCP/IP connections (available file descriptors) is exceeded. Default is 90 percent of the available file descriptors.					
PSTORE	NO | HOT | COLD	O	z	u	w	b
	Defines the status of the persistent store at broker startup, including the condition of persistent units of work (UOWs). With any value other than NO, PSTORE-TYPE must be set. NO No persistent store. HOT Persistent UOWs are restored to their prior state during initialization. COLD Persistent UOWs are not restored during initialization, and the persistent store is considered empty. **Note:** For a hot or cold start, the persistent store must be available when your broker is restarted.					
PSTORE-REPORT	NO | YES	O	z	u	w	b
	Determines whether PSTORE report is created.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	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 (Linux, Windows) | ADABAS (all platforms)	O	z	u	w	b
	Describes the type of persistent store driver required. DIV Data in Virtual. z/OS only, and default on this platform. CTREE c-tree database. Linux and Windows only. See ADABAS Adabas. All platforms. See also ***Adabas-specific Attributes*** (below) and *Managing the Broker Persistent Store* in the platform-specific Administration documentation.					
PSTORE-VERSION	5	O	z	u	w	b
	Determines the version of the persistent store. PSTORE-VERSION=5 is the only supported version since EntireX version 10.8. **Note:** To change the value of PSTORE-VERSION, the persistent store must be empty (all units of work must be consumed). If the persistent store is not empty, the start of the Broker with a changed PSTORE-VERSION may fail with error ETBE0741 or ETBM0745.					
SECURITY	NO | YES	O	z	u	w	b
	Determines whether EntireX Security is activated. NO EntireX Security is not activated. YES EntireX Security is activated. See *EntireX Security*.					
SERVER-DEFAULT	*n* | UNLIM	O	z	u	w	b
	Default number of servers that are allowed for every service. *n* Number of servers. UNLIM The number of servers is restricted only by the number of servers globally available. Precludes the use of NUM-SERVER=AUTO. This value can be overridden by specifying a SERVER-LIMIT for the service. A value of 0 (zero) is invalid.					
SERVICE-UPDATES	YES | NO	O	z	u	w	b
	Switch on/off the automatic update mode of the broker.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	`YES` The broker reads the attribute file whenever a service registers for the first time. This allows the broker to honor modifications in the attribute file *without* a restart. The attribute file is read only when the first server registers for a particular service; it is not reread when a second replica is activated. `NO` The attribute file is read only once during broker startup. Any changes to the attribute file will be honored only if the broker is restarted.					
`SHORT-BUFFER-DEFAULT`	`UNLIM` | *n*	O	z	u	w	b
	Number of short buffers to be allocated for each service. `UNLIM` The number of short message buffers is restricted only by the number of buffers globally available. Precludes the use of `NUM-SHORT-BUFFER=AUTO`. *n* Number of buffers. This value can be overridden by specifying a `SHORT-BUFFER-LIMIT` for the service. A value of 0 (zero) is invalid.					
`STORAGE-REPORT`	`NO` | `YES`	O	z	u	w	b
	Create a storage report about broker memory usage. `NO` Do not create the storage report. `YES` Create the storage report. See *Storage Report*.					
`STORE`	`OFF` | `BROKER`	O	z	u	w	b
	Sets the default `STORE` attribute for all units of work. This attribute can be overridden by the `STORE` field in the Broker ACI control block. `OFF` Units of work are not persistent. `BROKER` Units of work are persistent.					
`TRACE-LEVEL`	`0`-`4`	O	z	u	w	b
	The level of tracing to be performed while the broker is running. `0` No tracing. Default value. `1` Traces incoming requests, outgoing replies, resource usage and conversion errors. `2` All of trace level 1, plus all main routines executed. `3` All of trace level 2, plus all routines executed. `4` All of trace level 3, plus Broker ACI control block displays. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use Command Central or the EntireX Broker command-line utility ETBCMD.					
TRANSPORT	TCP-NET | TCP | SSL | NET	O	z			b
	TCP | SSL	O		u	w	
	The broker transport may be specified as any combination of one or more of the following methods: TCP TCP/IP is supported. SSL SSL/TLS is supported. NET Entire Net-Work is supported. This value is not supported for a broker under Linux or Windows. **Examples:** TRANSPORT=NET specifies that only the Entire Net-Work transport method will be supported by the broker. TRANSPORT=TCP-NET specifies that both the TCP/IP and Net-Work transport methods will be supported by the broker. TRANSPORT=TCP-SSL-NET specifies that the TCP/IP, SSL/TLS, and Entire Net-Work transport methods will be supported by the broker. The parameters for each transport method are described in the respective section: **TCP** | **SSL** | **NET**.					
TRAP-ERROR	*nnnn*	O	z	u	w	b
	Where *nnnn* is the four-digit API error number that triggers the trace handler, for example 0007 (Service not registered). Leading zeros are not required. There is no default value. See *Deferred Tracing* under z/OS | Linux | Windows in the platform-specific Administration documentation.					
TRBUFNUM	*n*	O	z	u	w	b
	Changes the trace to write trace data to internal trace buffers. *n* is the size of the trace buffer in 64 KB units. There is no default value.					
TRMODE	WRAP	O	z	u	w	b
	Changes the trace mode. WRAP is the only possible value. This value instructs broker to write the trace buffer (see TRBUFNUM) if an event occurs. This event is triggered by a matching TRAP-ERROR during request processing or when an exception occurs.					
UMSG	See MAX-MESSAGES-IN-UOW.					
UOW-DATA-LIFETIME	1D | *n*S | *n*M | *n*H | *n*D	O	z	u	w	b
	Defines the default lifetime for units of work for the service.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	*n*S Number of seconds the UOW can exist (max. 2147483647). *n*M Number of minutes the UOW can exist (max. 35791394). *n*H Number of hours the UOW can exist (max. 596523). *n*D Number of days the UOW can exist (max. 24855). If the UOW is inactive - that is, is not processed within the time limit - it is deleted and given a status of TIMEOUT. This attribute can be overridden by the UWTIME field in the Broker ACI control block. See *Timeout Considerations for EntireX Broker*.					
UOW-MSGS	See MAX-MESSAGES-IN-UOW.					
UOW-STATUS-LIFETIME	no value | *n*[S] | *n*M | *n*H | *n*D	O	z	u	w	b
	The value to be added to the UOW-DATA-LIFETIME (lifetime of associated UOW). If a value is entered, it must be 1 or greater; a value of 0 will result in an error. If no value is entered, the lifetime of the UOW *status* information will be the same as the lifetime of the UOW itself. *n*S Number of seconds the UOW status exists longer than the UOW itself (max. 2147483647). *n*M Number of minutes (max. 35791394). *n*H Number of hours (max. 596523). *n*D Number of days (max. 24855). This attribute is ignored if PSTORE=NO is defined. The lifetime determines how much additional time the UOW status is retained in the persistent store and is calculated from the time at which the associated UOW enters any of the following statuses: PROCESSED, TIMEOUT, BACKEDOUT, CANCELLED, DISCARDED. The additional lifetime of the UOW status is calculated only when broker is executing. Value in UOW-STATUS-LIFETIME supersedes the value (if specified) in attribute UWSTATP. **Note:** If no unit is specified, the default unit is seconds. The unit does not have to be identical to the unit specified for UOW-DATA-LIFETIME.					
UWSTAT-LIFETIME	Alias for UOW-STATUS-LIFETIME.					
UWSTATP	0 | *n*	O	z	u	w	b
	Contains a multiplier used to compute the lifetime of a persistent status for the service. The UWSTATP value is multiplied by the UOW-DATA-LIFETIME value (the lifetime of the associated UOW) to determine the length of time the status will be retained in the persistent store. 0 The status is not persistent.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	1-254 Multiplied by the value of UOW-DATA-LIFETIME to determine how long a persistent status will be retained. **Note:** This attribute has not been supported since EntireX version 7.3. Use UOW-STATUS-LIFETIME instead.					
UWTIME	Alias for UOW-DATA-LIFETIME.					
WAIT-FOR-ACTIVE-PSTORE	NO | YES	O	z	u	w	b
	Determines whether broker should wait for the Adabas Persistent Store to become active, or until c-tree PSTORE files become available. NO If broker should start with a PSTORE-TYPE=ADABAS and the database is not active or is not accessible, broker will stop. If broker should start with a PSTORE-TYPE=CTREE and the c-tree files are still in use, broker will stop. YES If broker should start with a PSTORE-TYPE=ADABAS and the database is not active or is not accessible, broker will retry every 10 seconds to initiate communications with the PSTORE. Broker will reject any user requests until it is able to contact the Adabas database. If broker should start with a PSTORE-TYPE=CTREE and the c-tree files are still in use, broker will retry every 10 seconds to rebuild the persistent data. Broker will reject any user requests until it is able to rebuild the persistent data.					
WORKER-MAX	64 | *n* (min. 1, max. 64)	O	z	u	w	b
	Maximum number of worker tasks the broker can use.					
WORKER-MIN	1 | *n* (min. 1, max. 64)	O	z	u	w	b
	Minimum number of worker tasks the broker can use.					
WORKER-NONACT	70S | *n* | *n*S | *n*M | *n*H	O	z	u	w	b
	Non-activity time to elapse before a worker tasks is stopped. *n* Same as *n*S. *n*S Non-activity time in seconds (default 70, max. 2147483647). *n*M Non-activity time in in minutes (max. 35791394). *n*H Non-activity time in hours (max. 596523). **Caution:** A value of 0 (zero) is invalid. If you set this value too low, additional overhead is required for starting and stopping worker tasks. The default and recommended value is 70S.					
WORKER-QUEUE-DEPTH	1 | *n* (min. 1)	O	z	u	w	b

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Number of unassigned user requests in the input queue before another worker task gets started. The default and recommended value is `1`. A higher value will result in longer broker response times.					
`WORKER-START-DELAY`	*`internal-value`* | *`n`*	O	z	u	w	b
	`n` Delay is extended by *`n`* seconds. Delay after a successful worker task invocation before another worker task can be started to handle current incoming workload. This attribute is used to avoid the risk of recursive invocation of worker tasks, because starting a worker task itself causes workload increase. If no value is specified, an internal value calculated by the broker is used to optimize dynamic worker management. This calculated value is the maximum time required to start a worker task.					

Service-specific Attributes

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

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

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

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

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

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

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

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	`YES` Store the EOC notifications if the server is not ready to receive and then notify the server if possible. If a server is not ready to receive an EOC notification, it can be stored or discarded. If it is stored, the server is notified, if possible, when it is ready to receive. **Caution:** The behavior activated by this parameter can be relied upon only during a single lifetime of the broker kernel. Specifically, conversations containing units of work, whose lifetime can span multiple broker kernel sessions, cannot be assumed to show this behavior, even with `NOTIFY-EOC=YES`.					
`NUM-UOW`	Alias for `MAX-UOWS`.					
`SERVER`	`A32` (case-sensitive)	R	z	u	w	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	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	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.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	B
	Note: Linux and Windows: This limit also includes any attach server you are using. Mal you increase the number by one for each attach server you use.					
SERVER-NONACT	5M | *n* | *n*S | *n*M | *n*H	R	z	u	w	
	Non-activity time for servers. A server that does not issue a broker request within the sp time limit is treated as inactive and all resources for the server are freed. *n* Same as *n*S. *n*S Non-activity time in seconds (max. 2147483647). *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 ta non-activity time for the server.					
SERVICE	A32 (case-sensitive)	R	z	u	w	
	Part of the name that identifies the service together with the CLASS and SERVER attribut CLASS must be specified first, followed immediately by SERVER and SERVICE. The SERVICE attribute names EXTRACTOR and DEPLOYMENT are reserved for Software A internal use and should not be used in customer-written applications. Valid characters for name are letters a-z, A-Z, numbers 0-9, hyphen and underscore. Do not use dollar, perc period or comma. See also the restriction for CLASS attribute names.					
SHORT-BUFFER-LIMIT	UNLIM | *n*	O	z	u	w	
	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 gl available. Precludes the use of NUM-SHORT-BUFFER=AUTO in the Broker section 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 sp or the SHORT-BUFFER-LIMIT attribute must be suppressed entirely for the service so th default (SHORT-BUFFER-DEFAULT) becomes active.					
STORE	OFF | BROKER	O	z	u	w	
	Sets the default STORE attribute for all units of work sent to the service. OFF Units of work are not persistent. BROKER Units of work are persistent. This attribute can be overridden by the STORE field in the Broker ACI control block.					
TRANSLATION	NO | *name* (A255)	O	z	u	w	

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	Activates translation user exit for character conversion. NO If translation is not to be used - for example for binary payload (broker messages) - either omit the TRANSLATION attribute or specify TRANSLATION=NO. *name* Name of Translation User Exit. See also *Configuring Translation User Exits* under *Configuring Broker for Internationalization* in the platform-specific Administration 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	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	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). *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.					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	B
	Note: If no unit is specified, the default unit is seconds. The unit does not have to be ide to the unit specified for UOW-DATA-LIFETIME.					
UWSTATP	0 | *n*	O	z	u	w	
	Contains a multiplier used to compute the lifetime of a persistent status for the service. UWSTATP value is multiplied by the UOW-STATUS-LIFETIME value (the lifetime of the ass 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 persi status will be retained. This attribute is ignored if PSTORE=NO is defined. **Note:** This attribute has not been supported since EntireX version 7.3. Use UOW-STATUS-LIFETIME instead.					
UWSTAT-LIFETIME	Alias for UOW-STATUS-LIFETIME.					
UWTIME	Alias for UOW-DATA-LIFETIME.					

Wildcard Service Definitions

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

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

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

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

Service Update Modes

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

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

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

OPTION Values for Conversion

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

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

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

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

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

Codepage-specific Attributes

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

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

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

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

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

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

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

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

Additional Notes

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

Adabas SVC/Entire Net-Work-specific Attributes

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`EXTENDED-ACB-SUPPORT`	`NO` | `YES`	O	z			b
	Determines whether extended features of Adabas version 8 (or above) are supported. `NO` No features of Adabas version 8 or above will be used. `YES` Informs broker kernel to provide Adabas/WAL version 8 transport capability. This parameter is required for sending/receiving more than 32 KB data over Adabas [NET] transport. This value should be set only if you have installed Adabas/WAL version 8, Adabas SVC, and included Adabas/WAL version 8 load libraries into the steplib of broker kernel; otherwise, unpredictable results can occur.					
`FORCE`	`NO` | `YES`	O	z			b
	Determines whether DBID table entries can be overwritten. `NO` Overwrite of DBID table entries not permitted. `YES` Overwrite of DBID table entries permitted. This is required when the DBID table entry is not deleted after abnormal termination. **Caution:** Overwriting an existing entry prevents any further communication with the overwritten node. Use `FORCE=YES` only if you are absolutely sure that no target node with that DBID is active.					
`IDTNAME`	*`idtname`*`(A8)` | `ADABAS5B`	O				b
	If an ID table name is specified with the appropriate `ADARUN` parameter for Entire Net-Work, Adabas or Natural, the same name must be specified here. The ID table is used to perform various internal functions, including communication between the caller program and the EntireX Broker. Only supported under BS2000.					
`IUBL`	`8000` | *`n`*	O	z			b
	This parameter sets the maximum length (in bytes) of the buffer that can be passed from the caller to EntireX Broker. The maximum size of `IUBL` is the same					

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	as the maximum value of the Adabas parameter LU. See the *Adabas Operations Manual*. IUBL must be large enough to hold the maximum send-length plus receive-length required for any caller program plus any administrative overhead for Adabas and Entire Net-Work control structures.					
LOCAL	NO | YES	O	z			b
	For remote nodes accessed via Entire Net-Work, the attribute LOCAL specifies whether the target ID defined with the NODE attribute can be accessed only locally, or also remotely. NO DBID is *global* and can be accessed from remote nodes via Entire Net-Work. YES DBID is *local* and cannot be accessed from remote nodes via Entire Net-Work.					
MAX-MESSAGE-LENGTH	2147483647 | *n*	O	z	u	w	b
	Maximum message size that the broker kernel can process using transport method NET. The default value represents the highest positive number that can be stored in a four-byte integer.					
NABS	10 | *n*	O	z			b
	The number of attached buffers to be used (max. 524287). An attached buffer is an internal buffer used for interprocess communication. An attached buffer pool equal to the NABS value multiplied by 4096 will be allocated. This buffer pool must be large enough to hold all data (IUBL) of all parallel calls to EntireX Broker. The following formula can be used to calculate the value for NABS: NABS = NCQE *IUBL / 4096.					
NCQE	10 | *n*	O	z			b
	NCQE defines the number of command queue elements which are available for processing commands arriving at the broker kernel over Adabas SVC / Net-Work transport mechanism. Sufficient NCQE should be allocated to allow this transport mechanism to process multiple broker commands concurrently. Each command queue element requires 192 bytes, and the element is released when either the user (client or server) has received the results of the command, or if the command is timed out. The number of command queue elements required to handle broker calls depends on the number of parallel active broker calls that are using the transport mechanism Adabas SVC / Entire Net-Work. For example, all broker commands issued by client or server components using this transport mechanism:					
NODE	1-65534	R	z			b
	Defines the unique DBID for EntireX Broker.					

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

Security-specific Attributes

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`ACCESS-SECURITY-SERVER`	`NO` | `YES`	O				b
	Determines where authentication is checked. `NO` Authentication is checked in the broker tasks. This requires broker to be running under TSOS in order to execute privileged security checks. `YES` Authentication is checked in the EntireX Broker Security Server for BS2000. This does not require broker to be running under TSOS. See ***EntireX Broker Security Server for BS2000***.					
`IGNORE-STOKEN`	`NO` | `YES`	O	z	u	w	b
	Determines whether the value of the ACI field `SECURITY-TOKEN` is verified on each call.					
`SECURITY-LEVEL`	`AUTHORIZATION` | `AUTHENTICATION`	O	z	u	w	b
	Specifies the mode of operation. `AUTHORIZATION` Authorization and authentication (not under BS2000). `AUTHENTICATION` Authentication. **Note:** In version 8.0, the default value for this parameter was `AUTHORIZATION`.					
`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(not applicable to z/OS and BS2000).					
`TRACE-LEVEL`	`0`-`4`	O	z	u	w	b
	Trace level for EntireX Security. It overrides the global value of trace level in the attribute file. `0` No tracing. Default value. `1` Log security violations and access denied/permitted.					

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

TCP/IP-specific Attributes

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`CONNECTION-NONACT`	*n* | *n*S | *n*M | *n*H	O	z	u	w	b
	Non-activity of the TCP/IP connection, after which a close is performed and the connection resources are freed. If this parameter is not specified here, broker will close the connection only when the application (or the network itself) terminates the connection. *n* Same as *n*S. *n*S Non-activity time in seconds (min. 600, max. 2147483647). *n*M Non-activity time in minutes (min. 10, max. 35791394). *n*H Non-activity time in hours (max. 596523). If not specified, the connection non-activity is not monitored by the broker. On the stub side, non-activity can be set with the environment variable `ETB_NONACT`. See *Limiting the TCP/IP Connection Lifetime* under z/OS | Linux | Windows | z/VSE in the platform-specific *Administering Broker Stubs* documentation. Both sides (the stub side and the broker kernel side) act independently of each other.					
`HOST`	`0.0.0.0` | *hostname* | *IP address*	O	z	u	w	b
	The address of the network interface on which broker will listen for connection requests. If `HOST` is not specified, broker will listen on any attached interface adapter of the system (or stack). A maximum of five `HOST/PORT` pairs can be specified to start multiple instances of broker's TCP/IP transport communicator.					
`MAX-MESSAGE-LENGTH`	`2147483647` | *n*	O	z	u	w	b
	Maximum message size that the broker kernel can process using transport method TCP/IP. The default value represents the highest positive number that can be stored in a four-byte integer.					
`PORT`	`1025-65535`	O	z	u	w	b
	The TCP/IP port number on which the broker will listen for connection requests.					

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

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	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.					
TRACE-LEVEL	0-4	O	z	u	w	b
	The level of tracing to be performed while the broker is running with transport method TCP/IP. It overrides the global value of trace level for all TCP/IP routines. 0 No tracing. Default value. 1 Display IP address of incoming request, display error number of outgoing error responses. 2 All of trace level 1, plus errors if request entries could not be allocated. 3 All of trace level 2, plus all routines executed. 4 All of trace level 3, plus function arguments and return values. Trace levels 2, 3 and 4 should be used only when requested by Software AG Support. If you modify the TRACE-LEVEL attribute, you must restart the broker for the change to take effect. For temporary changes to TRACE-LEVEL without a broker restart, use the EntireX Broker command-line utility ETBCMD.					

SSL/TLS-specific Attributes

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

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

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

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

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

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
	`NO` No client certificate required (default). For more information see *SSL/TLS, HTTP(S), and Certificates with EntireX*.					

Adabas-specific Attributes

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

Attribute	Values	Opt/ Req	Operating System			
			z/OS	Linux	Windows	BS2000
`BLKSIZE`	`126-20000`	O	z	u	w	b
	Optional blocking factor used for message data. If not specified, broker will split the message data into 2 KB blocks to be stored in Adabas records. The maximum value depends on the physical device assigned to data storage. See the *Adabas* documentation. For reasons of efficiency, do not specify a `BLKSIZE` much larger than the actual total size of the UOW data to be written. The total UOW size is the sum of all messages in the UOW plus 41 bytes of header information. This takes effect only after COLD start. The `BLKSIZE` parameter applies only for a cold start of broker; subsequently the value of `BLKSIZE` is taken from the last cold start. Default value is 2000.					
`DBID`	`1-32535`	R	z	u	w	b
	Database ID of Adabas database where the persistent store resides.					
`FNR`	`1-32535`	R	z	u	w	b
	File number of broker persistent store file.					
`FORCE-COLD`	`N` | `Y`	O	z	u	w	b
	Determines whether a broker cold start is permitted to overwrite a persistent store file that has been used by another broker ID and/or platform. Specify `Y` to allow existing information to be overwritten.					
`MAXSCAN`	`0` | *n*	O	z	u	w	b
	Limits display of persistent UOW information in the persistent store through Command and Information Services. Default value is 1000.					
`OPENRQ`	`N` | `Y`	O	z	u	w	b
	Determines whether driver for Adabas persistent store is to issue an `OPEN` command to Adabas.					
`TRACE-LEVEL`	`0-4`	O	z	u	w	b
	Trace level for Adabas persistent store. It overrides the global value of trace level in the attribute file.					

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

Application Monitoring-specific Attributes

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

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

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.

4 Configuring Broker for Internationalization

- Configuring ICU Conversion 64
- Building and Installing ICU Custom Converters 65
- Writing Translation User Exits 65
- Configuring Translation User Exits 67
- Configuring Translation 68

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`. Example:

- ICU Conversion with SAGTCHA for *ACI-based Programming*, *RPC-based Components* [(1)] and *Direct RPC* [(1)]:

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

Note: [(1)]Conversion with multibyte, double-byte and other complex codepages for RPC-based components and Reliable RPC is not supported on BS2000.

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 converters (codepages) are not supported under BS2000.

Writing Translation User Exits

This section covers the following topics:

- Introduction
- Structure of the TRAP Control Block
- Using the TRAP Fields

Introduction

EntireX Broker provides an interface to enable user-written translation routines in the programming language Assembler. 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 Assembler dummy section TR$TRAP covers the layout of the TRAP control block:

```
TR$TRAP  DSECT ,
TR$TYPE  DS       F            TRAP type
TR$TYP2  EQU      2            TRAP type ETB 121
TR$ILEN  DS       F            Input buffer length
TR$IBUF  DS       A            Address of input buffer
TR$OLEN  DS       F            Output buffer length
TR$OBUF  DS       A            Address of output buffer
TR$DLEN  DS       F            Length of data returned:
*                              Should be set to the minimum value of TR$ILEN
*                              and TR$OLEN.
TR$SHOST DS       F            Sender's host:
*                              x'00000000' = little endian
*                              x'00000001' = big endian
TR$SCODE DS       F            Sender's character set:
SEBCIBM  EQU      X'00000022'  EBCDIC (IBM)
SEBCSNI  EQU      X'00000042'  EBCDIC (SNI)
SA88591  EQU      X'00000080'  ASCII
TR$RHOST DS       F            Receiver's host     --> see TR$SHOST
TR$RCODE DS       F            Receiver's char set --> see TR$SCODE
TR$BHOST DS       F            BROKER host         --> see TR$SHOST
TR$BCODE DS       F            BROKER char set     --> see TR$SCODE
TR$SENVA DS       F            Sender's ENVIRONMENT field supplied:
OFF      EQU      X'00000000'  ENVIRONMENT field not set
ON       EQU      X'00000001'  ENVIRONMENT field set
*
TR$RENVA DS       F            Receiver's ENVIRONMENT field supplied:
*                              --> see TR$SENVA
TR$SENV  DS       CL32         Sender's ENVIRONMENT field
TR$RENV  DS       CL32         Receiver's ENVIRONMENT field
TR$LEN   EQU      *-TR$TRAP    Length of 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 module must be accessible to the Broker worker threads.

1 Copy the user-written translation module into the EntireX Broker load library (EXX103.LIB).

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

```
TRANSLATION=MYTRANS
```

Configuring Translation

To configure translation

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

```
TRANSLATION=SAGTCHA
```

5 Managing the Broker Persistent Store

- Implementing an Adabas Database as Persistent Store 70

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 BS2000, the broker persistent store can be implemented with the Adabas database of Software AG. This chapter covers the following topics:

See also *Concepts of Persistent Messaging*.

Implementing an Adabas Database as Persistent Store

- Introduction
- Adabas Persistent Store Parameters
- Configuring and Operating the Adabas Persistent Store
- Adabas DBA Considerations

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, Linux, Windows and BS2000, and can be used where the database resides on a different machine to the 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

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
- Table of Adabas Parameter Settings
- Estimating the Number of Records to be Stored
- Estimating the Number of Records to be Stored
- Tips on Transports, Platforms and Versions

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.
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.
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

Topic	Description
	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.

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
■ 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)	■ = 164,548 * (96 + 3 + 3 + 1) ■ = 16,948,517 bytes

Calculation Factors	Required Space
■ (4,092 ASSO block 10% padding)	■ = 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.

6 Broker Resource Allocation

- General Considerations 78
- Specifying Global Resources 78
- Restricting the Resources of Particular Services 79
- Specifying Attributes for Privileged Services 81
- Maximum Units of Work 81
- Calculating Resources Automatically 82
- Dynamic Memory Management 84
- Dynamic Worker Management 85
- Storage Report 87
- Maximum TCP/IP Connections per Communicator 88

The EntireX Broker is a multithreaded application and communicates among multiple tasks in memory pools.

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, that is, 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.

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

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

Creating a Storage Report

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

Platform-specific Rules

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

Sample Storage Report

The following is an excerpt from a sample STORAGE report.

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

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

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

Maximum TCP/IP Connections per Communicator

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

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

With the Broker-specific attribute `POLL`, these restrictions can be lifted under z/OS and Linux. 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 ACI Programming documentation.

7 EntireX Broker Security Server for BS2000

- Activating Authentication 90
- Starting the Broker Security Server 90
- Stopping the Broker Security Server 91
- Tracing with the Broker Security Server 91
- Broker Security Server Parameters 92

The Broker Security Server authenticates users who log on to EntireX Broker, e.g. it performs a user ID and password check against the operating system. The user ID must exist under BS2000. Since the server reads information from the user catalog, it requires administrator rights at runtime. The Broker Security Server task therefore needs to run under a privileged user ID (TSOS).

User IDs and passwords are case-insensitive.

The Broker Security Server can handle multiple broker instances on BS2000.

Activating Authentication

To activate authentication, switch on security in the broker attribute file. Add the following two parameters to `ETB-ATTR` or, if you use the delivered attribute file, switch the `SECURITY` parameter to "YES".

```
SECURITY =YES
ACCESS-SECURITY-SERVER=YES
```

The Broker Security Server requires administrator rights and must be run under a privileged user ID. Set up the correct broker load library in `START-SECURITY-SERVER`.

Note: If `ACCESS-SECURITY-SERVER` is set to "NO", EntireX Broker itself will do the authentication. In that case EntireX Broker must run under a privileged user ID and the Broker Security Server is not needed.

Starting the Broker Security Server

To start the Broker Security Server

1 Set up the correct broker library within `START-SECURITY-SERVER`, because the server task does not usually run under same user ID where the module library resides.

2 Issue the following command from a privileged user ID (TSOS) to run the server:

```
/ENTER-PROCEDURE *LIB(LIB=$kkk.EXXnnn.JOBS,    -
/                 ELE=START-SECURITY-SERVER),  -
/                 JOB-NAME=SECUSERV,LOG=*NO
```

where `$kkk` is the user ID under which the broker library resides.

Stopping the Broker Security Server

To stop the Broker Security Server from a privileged user ID

- Enter:

```
/INFORM-PROGRAM MSG='EOJ',JOB-IDENTIFICATION=*TSN(TSN=tsn)
```

  where <*tsn*> is the BS2000 task number associated with the server.

To stop the Broker Security Server from an operator console

- Enter:

```
/INTR tsn,EOJ
```

  where *tsn* is the BS2000 task number associated with the server.

To stop the Broker Security Server from a non-privileged user ID

- Enter the following SDF command:

```
/CALL-PROCEDURE (EXX103.JOBS, STOP-SECURITY-SERVER)
```

  Note: This works from all user IDs in the system.

Tracing with the Broker Security Server

The Broker Security Server comes with a trace facility that can be used to track the IDs of users logging on to EntireX Broker. It also produces some diagnostic messages that are helpful for problem analysis. By default, no tracing is performed.

To switch on tracing for Broker Security Server

- Set up an `SDF` variable in the server's job control.

```
TRACE='ON'
```

To switch off tracing for Broker Security Server

- Set the following in the server's job control.

```
TRACE='OFF'
```

See *EntireX Broker Security Server for BS2000*.

Broker Security Server Parameters

The Broker Security Server uses a global common memory pool for communicating with its clients (broker instances). This common memory pool is established and initialized by the Broker Security Server task. If the pool already exists, the Broker Security Server will not start. This is possible if a client did not disconnect correctly or the Broker Security Server is already running. The message SECE010 “Broker Security Server already active” message is issued. In such a situation, you can use the `FORCE` parameter to reconnect the Broker Security Server to the security common memory pool. The pool is initialized again, and open requests are deleted, which means that authentication for these clients will fail. The initialization process is indicated by message SECI004 “Running with FORCE = YES, the security CMP will be newly initialized”.

Before you set `FORCE=YES`, make sure that no other Broker Security Server is running.

```
FORCE='NO/YES'
```

8 Administering Broker Stubs

- Available Stub 94
- Linking the Stubs 94
- Transport Methods for Broker Stubs 97
- Using Job Variables 100
- Using BROKER under openUTM 100

Available Stub

The stub BROKER is available under BS2000. It is used with the programming languages Assembler | C | COBOL | Natural | PL/I.

It can be used in all environments that use batch or Dialog (formerly TIAM). Supported transport methods are NET and TCP/IP.

Using transport method NET will greatly improve performance when running Broker kernel and applications on the same machine. We recommend using the transport method NET for all local communication within BS2000. In order to use the transport method NET for messages involving more than 32 KB, you must install cross-memory services of Adabas 8 or above. If you have not yet installed Adabas 8 cross-memory services, you can instead use TCP/IP to transport more than 32 KB of data.

When using Adabas 8 or above with any of the BS2000 stubs to transport more than 32 KB of data, note the following:

- Adabas/WAL 8 or above must be installed.
- The Adabas/WAL 8 or above link routine must be used by the application or TP monitor.
- Adabas/WAL 8 or above libraries must be used by the Broker kernel.
- Adabas/WAL 8 or above libraries must be used by the Broker stubs.
- The parameter EXTENDED-ACB-SUPPORT must be used for transmitting data from Adabas (NET).
- Sufficient buffer space by IUBL, NABS and NUM-COMBUF must be specified.

Linking the Stubs

This section describes how to link the stubs:

- Stub BROKER

- Stub BROKER with Natural

Stub BROKER

To prepare your application to perform Broker calls

1 Link the front-end module BROKER from the EntireX load library (EXX103.LIB) to your application. It has the entry point "BROKER". When BROKER is first called, it loads the actual stub module from the EntireX load library and transfers control to it.

2 Add the following assignment to the startup procedure:

```
/ADD-FILE-LINK LINK-NAME=ETBLIB,FILE-NAME=<EXX_load_library>
```

3 To enable the Adabas transport method, add the following assignment to the startup procedure:

```
/ADD-FILE-LINK LINK-NAME=DDLIB,FILE-NAME=<adabas_load_library>
```

As a result, the required Adabas link module is loaded from the appropriate Adabas load library.

4 Add the following assignment to the startup procedure:

```
/ADD-FILE-LINK LINK-NAME=DDLNKPAR,FILE-NAME=<adalnk-parameter>
```

As a result, ADAUSER reads the configuration parameters, for example IDTNAME.

Stub BROKER with Natural

To prepare your application to perform Broker calls

1 Add the following assignment to the startup procedure:

```
/ADD-FILE-LINK LINK-NAME=BLSLIB00,FILE-NAME=<EXX_load_library>
```

2 Add the following assignment to the startup procedure:

```
/ADD-FILE-LINK LINK-NAME=ETBLIB,FILE-NAME=<EXX_load_library>
```

3 Start Natural with the following profile parameters:

```
RCA=(BROKER),RCALIAS=(BROKER,BROKER)
```

As a result, BROKER is loaded dynamically, and each broker call will use this stub.

Note:
This dynamic load/execute will work even if an old NATETB23 has already been linked to the shared Natural nucleus as static module. You need not link BROKER statically to the Natural front-end. It is, however, possible to link BROKER statically to the front-end Natural and remove the NATETB23 module from the shared Natural nucleus to avoid specifying the profile parameters mentioned above.

Transport Methods for Broker Stubs

- Transport Method Values
- Using Transport Methods
- Setting the Timeout for the Transport Method
- Tracing for Broker Stubs

Transport Method Values

The following table describes the possible values for the transport methods. Default is NET.

Transport Value	Description
NET	Default. Use Adabas BS2000 Communication Environment as transport method. It is also possible to communicate remotely with the transport method NET from an application (client or server) to the broker kernel using Entire Net-Work. For remote NET communication, Entire Net-Work must be installed both on the machine where the broker kernel runs and on the machine where your application (client or server) runs, and a connection must be established. Using Adabas/WAL V8 allows more than 32 KB of data to be communicated. Otherwise the following maximum values are allowed: **ACI Version** — **Max Send/Receive length** 1 — 32167 2,3 — 31647 4-8 — 31643 9 or above — 31123 **Note:** If Adabas version 8 or above is *not* used, these same limits still apply.
TCP	Use TCP/IP as transport method.

Using Transport Methods

This section covers specifications for transport methods as part of the broker ID.

Note: If no transport method has been specified as part of the broker ID, default value NET is used.

Using Adabas Communication

To Use Adabas Communication as Transport Method

- Specify:

```
broker-id::NET
```

Notes:

1. Port number does not apply and is therefore left blank. Adabas communication is the transport method.
2. It is not possible to provide the `IDTNAME` with the broker ID. The `IDTNAME` is specified in a parameter file controlled by the `ADAUSER` module (assigned using link name `DDLNKPAR`).

- **Using TCP/IP**

To use TCP/IP as transport method

- Specify:

```
broker-id:nnnnn:TCP
```

where *nnnnn* is a placeholder for a port number.

Setting the Timeout for the Transport Method

Introduction

If the transport layer is interrupted, communication between the broker and the stub - that is, client or server application - is no longer possible. A client or server might possibly wait infinitely for a broker reply or message in such a situation. To prevent this and return control to your calling application in such a situation, set a timeout value for the transport method.

The timeout settings for transport layers are independent of the timeout settings of the broker.

Setting the timeout for the transport layer is possible for the transport method TCP, and is supported by broker stub `BROKER`.

Transport Timeout Values

The timeout value for the transport method is set by the environment variable `ETB_TIMEOUT` on the stub side. This transport timeout is used together with the broker timeout - which is set by the application in the `WAIT` field of the broker ACI control block - to calculate the actual value for the transport layer's timeout. The following table describes the possible values for the transport timeout:

Transport Timeout Value	Description
0	Infinite wait for the application.
n	The transport method additionally waits this time in seconds. A negative value is treated as `TIMEOUT=0` (infinite wait for the application).
nothing set	Transport method waits additional 20 seconds.

The actual timeout for transport layer equals broker timeout (`WAIT` field) + timeout value for transport method.

Tracing for Broker Stubs

Scope

Setting tracing is supported by the broker stub `BROKER` if transport method TCP is used. The stub tries to access the SDF variable `ETB-STUBLOG` (or, failing that, a job variable with the same name), to evaluate the value of the logging level. If the logging level is set, a sequential file will be created with the file name *9999.ETB* where *9999* is the task sequence number of the running task.

Trace	Level	Description
0	`NONE`	No tracing. Switch tracing off.
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 information from the transports.
3	`SUPPORT`	This is full tracing through the stub, including detailed traces of control blocks, message information, etc.

Activating Logging

To activate logging

- Set `JV ETB-STUBLOG` to value [1|2|3].

Where "1" is the lowest log level and "3" is the highest.

Using Job Variables

SDF and job variables (environment variables in an Open Systems architecture such as Linux or Windows) are used with the stub `BROKER` to read configuration parameters. BS2000 uses the hyphen character whereas environment variables use the underscore character. The stub attempts to read the SDF variable. If this fails, the job variable is read. If neither an SDF variable nor a job variable is read, it is assumed not using any environment variables.

Using BROKER under openUTM

You cannot use `BROKER` with dialog transactions under openUTM. You can, however, use `BROKER` within asynchronous transaction processing under openUTM. Prepare your Natural/UTM application as follows:

1. Link module `BROKER` from the EntireX library *EXX103.LIB* to the front-end part of your Natural/UTM application.
2. Add the following assignment to the Natural/UTM startup job:

```
/ADD-FILE-LINK LINK-NAME=ETBLIB,FILE-NAME=EXX_load_library
```

3. To enable the Adabas transport method, add the following assignment to the Natural/UTM startup job:

```
/ADD-FILE-LINK LINK-NAME=DDLIB,FILE-NAME=adabas_load_library
```

For more information on writing an asynchronous Natural/UTM transaction see section *Asynchronous Transaction Processing under UTM* in the Natural/UTM documentation.

9 Broker Command-line Utilities

- ETBINFO 102
- ETBCMD 108

EntireX Broker provides the following internal services: Command Service and Information Service, which can be used to administer and monitor brokers. 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
- Command-line Parameters
- Profile
- Format String

Running the Command-line Utility

In a BS2000 environment, run the command-line utility `ETBINFO` as shown below:

```
/CALL-PROCEDURE (LIB=EXX103.JOBS,ELE=ETBINFO)
```

This executes the utility in BS2000 dialog mode.

The `ETBINFO` parameters are supplied using an SDF variable, for example:

```
/COMMAND = '-b&(BROKER-ID) -dBROKER'
```

See the delivered `ETBINFO` job control in EXX103.JOBS.

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.
-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.

Option	Command-line Parameter	Req/ Opt	Explanation
-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.
-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.
-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.

Option	Command-line Parameter	Req/ Opt	Explanation
-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.

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
- CONV
- NET
- POOL
- PSF
- PSFADA
- PSFDIV
- RESOURCE
- 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.

Location of Profiles

On BS2000, the profiles used to control the amount of data displayed are contained in *EXX103.JOBS* and are called PRO-BROKER, PRO-CLIENT etc.

Example

Profile for object SERVICE: PRO-SERVICE.

To use a profile, the profile itself needs to be extracted from LMS library EXX103.JOBS. Uncomment the LMS-section including SDF variable COMMAND in S-procedure ETBINFO and adapt the profile name. For example:

```
...
/ START-LMS
//  MOD-LMS-DEFAULTS MAX-ERROR-WEIGHT=*RECOVERABLE
//  EXTRACT-ELEMENT                                                -
//     *LIB(LIB  = &(EXX-JOBS),                                    -
//          ELEM = PRO-SERVICE,                                    -
//          TYPE = S),                                             -
//          TO-FILE    = #PROFILE,                                 -
//          FILE-ATTR  = (ACCESS-METHOD=*SAM),                     -
//          WRITE-MODE = *ANY
//END
/COMMAND = '-b&(BROKER-ID) -dSERVICE -p#PROFILE -lFULL'
...
```

See also the delivered `ETBINFO` job control in EXX103.JOBS for more details.

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:

```
/COMMAND = '-b&(BROKER-ID) -d BROKER -f "%12.12CPLATNAME   %NUM-SERVER      ↩
%NUM-CLIENT"'
```

which produces the following output, for example:

```
BS2000 W 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:

```
/COMMAND = '-b&(BROKER-ID) -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:

```
/COMMAND = '-b&(BROKER-ID) -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
...
```

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
- Command-line Parameters
- List of Commands and Objects
- Examples

Running the Command-line Utility

In a BS2000 environment, run the `ETBCMD` command-line utility like this:

```
/CALL-PROCEDURE (LIB=EXX103.JOBS,ELE=ETBCMD)
```

This executes the utility in BS2000 dialog mode.

The ETBCMD parameters are supplied using an SDF variable. For example:

```
/COMMAND = '-b&(BROKER-ID) -dBROKER -cTRACE-ON -oLEVEL1'
```

See also delivered `ETBCMD` job control in EXX103.JOBS.

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` ■ `CLEAR-CMDLOG-FILTER` ■ `CONNECT-PSTORE` ■ `DISABLE-ACCOUNTING` ■ `DISABLE-CMDLOG-FILTER` ■ `DISABLE-CMDLOG` ■ `DISABLE-DYN-WORKER` ■ `DISCONNECT-PSTORE` ■ `ENABLE-ACCOUNTING`	R	Command to be performed. See ***List of Commands and Objects*** below.

Command-line Parameter	Option	Parameter	Req/ Opt	Explanation
		■ ENABLE-CMDLOG-FILTER ■ ENABLE-CMDLOG ■ ENABLE-DYN-WORKER ■ FORBID-NEWUOWMSGS ■ PING ■ PRODUCE-STATISTICS ■ PURGE ■ RESET-USER ■ RESUME ■ SET-CMDLOG-FILTER ■ 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.
	-e	errornumber	O	Error number being trapped.
	-E		O	Exclude attach servers from service shutdown.
help	-h		O	Prints help information.
class/server/service	-n	class/server/service	O	Service triplet.

Command-line Parameter	Option	Parameter	Req/ Opt	Explanation
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.
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.

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

Transport ID	Explanation
T04	TCP/IP communicator 5

List of Commands and Objects

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

Command	Object							
	BROKER	CONVERS-ATION	PARTICI-PANT	PSF	SECURITY	SERVER	SERVICE	TRANSPORT
ALLOW-NEWUOWMSGS				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							
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 system z/OS only.

Examples

Example	Description
`/COMMAND ='-h'`	Displays ETBCMD help text.
`/COMMAND='-b &(BROKER-ID) -d BROKER -c TRACE-OFF'`	Turns Broker tracing off.
`/COMMAND='-b &(BROKER-ID) -d BROKER -c TRACE-ON -o LEVEL2'`	Sets Broker trace level to 2.
`/COMMAND='-b &(BROKER-ID) -d BROKER -c SHUTDOWN'`	Performs Broker shutdown.
`/COMMAND='-b &(BROKER-ID) -d SERVICE -c SHUTDOWN -o IMMED -n ACLASS/ASERVER/ASERVICE'`	Shut down service `CLASS=ACLASS,SERVER=ASERVER,SERVICE=ASERVICE`. See also `SHUTDOWN SERVICE` under *Broker Command and Information Services* in the EntireX Broker documentation 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`.
`/COMMAND='-b &(BROKER-ID) -d SERVER -l FULL -f"%USER-ID %SEQNO"'`	1. Determine a list of all servers with sequence numbers.
`/COMMAND='-b &(BROKER-ID) -d SERVER -c SHUTDOWN -o IMMED -S32'`	2. Shutdown server with sequence number 32.
`/COMMAND='-b &(BROKER-ID) -d BROKER -c PING'`	Performs an EntireX ping against the Broker.
`/COMMAND='-b &(BROKER-ID) -d PSF -c DISCONNECT-PSTORE'`	Disconnects the Broker PSTORE.
`/COMMAND='-b &(BROKER-ID) -d PSF -c CONNECT-PSTORE'`	Connects the Broker PSTORE.
`/COMMAND='-b &(BROKER-ID) -d PSF -c PURGE -u 100000000U00001A'`	Purges a unit of work.
`/COMMAND='-b &(BROKER-ID) -d PSF -c ALLOW-NEWUOWMSGS'`	Allows new units of work to be stored.
`/COMMAND='-b &(BROKER-ID) -d PSF -c FORBID-NEWUOWMSGS'`	Disallows new units of work to be stored.

10 Operator Commands

- Command Syntax 116
- General Broker Commands 116
- Participant-specific Commands 121
- Security-specific Commands 126
- Transport-specific Commands 127
- XCOM-specific Commands 131

Command Syntax

The following command format is required to communicate with EntireX Broker, using the operator console. Parameters in UPPERCASE must be typed “as is”. Parameters in lowercase must be substituted with a valid value. Operator commands have the following format:

```
/INTR tsn,command[parameter]
```

where	*tsn*	is the BS2000 task sequence number of the EntireX Broker main task
	command	is the operator command
	parameter	is an optional parameter allowed by the operator command you are issuing

General Broker Commands

The following broker commands are available:

- BROKER TRACE
- DPOOL
- DRES
- DSTAT
- ETBEND
- ETBSTOP
- FLUSH
- PSTORE TRACE
- SHUTDOWN class,server,service
- TRACE
- TRAP-ERROR

BROKER TRACE

Alias of broker command `TRACE`. Modifies the setting of the broker-specific attribute `TRACE-LEVEL`.

Example

To set a trace level 2 for broker

- Enter command:

```
/INTR tsn,BROKER TRACE=2
```

If the console prompt is suppressed, enter an `MSG` command before the console command:

```
MSG partition_id
```

See `TRACE-LEVEL` under *Broker-specific Broker Attributes*.

DPOOL

Lists all memory pools currently allocated by EntireX Broker. Start address, pool size in bytes and name of pool are provided. There can be multiple entries for a specific type of pool.

Sample Output

```
ETBM0720 Operator typed in: DPOOL
ETBM0657 Broker pool usage:
ETBM0657 0x2338FFB8   16781380 bytes COMMUNICATION POOL
ETBM0657 0x243A9EB8     368964 bytes CONVERSATION POOL
ETBM0657 0x24404F38     233668 bytes CONNECTION POOL
ETBM0657 0x2443EF38    4395204 bytes LONG MESSAGES POOL
ETBM0657 0x24870BB8    3703876 bytes SHORT MESSAGES POOL
ETBM0657 0x24BF9398     134244 bytes PARTICIPANT POOL
ETBM0657 0x24C1AF78      36996 bytes PARTICIPANT EXTENSION POOL
ETBM0657 0x24C24798      26724 bytes PROXY QUEUE POOL
ETBM0657 0x24C2BDA8     131668 bytes SERVICE ATTRIBUTES POOL
ETBM0657 0x24C4CB98      54372 bytes SERVICE POOL
ETBM0657 0x24C5AF78      32900 bytes SERVICE EXTENSION POOL
ETBM0657 0x24C63B18      87268 bytes TIMEOUT QUEUE POOL
ETBM0657 0x24C79398     179300 bytes TRANSLATION POOL
ETBM0657 0x24CA5F38     176324 bytes UNIT OF WORK POOL
ETBM0657 0x24CD1798     391268 bytes WORK QUEUE POOL
ETBM0582 Function completed
```

DRES

Displays EntireX Broker's resource usage for conversations, message buffers, participants, services, the timeout queue, units of work, and the work queue. Resource usage provides the total number, the number of free elements, and the number of used elements.

Sample Output

```
ETBM0720 Operator typed in: DRES
ETBM0581 Broker resource usage:
ETBM0581 Resource ----------------- Total # --- Free # --- Used #
ETBM0581 Conversations                  4096        852       3244
ETBM0581 Long message buffers              0          0          0
ETBM0581 Short message buffers          8192       7384        808
ETBM0581 Participants                    256        235         21
ETBM0581 Services                        256        240         16
ETBM0581 Timeout Queue                  1280        845        435
ETBM0581 Units Of Work                     0          0          0
ETBM0581 Work Queue                      256        239         17
ETBM0582 Function completed
```

DSTAT

Displays the total number of active elements, and an optional high watermark for services, clients, servers, conversations and message buffers.

Sample Output

```
ETBM0720 Operator typed in: DSTAT
ETBM0580 Broker statistics:
ETBM0580 NUM-SERVICE ............... 0
ETBM0580 Services active ........... 7
ETBM0580 NUM-CLIENT ................ 0
ETBM0580 Clients active ............ 10
ETBM0580 Clients active HWM ........ 10
ETBM0580 NUM-SERVER ................ 0
ETBM0580 Servers active ............ 10
ETBM0580 Servers active HWM ........ 10
ETBM0580 NUM-CONVERSATION .......... 0
ETBM0580 Conversations active ...... 607
ETBM0580 Conversations active HWM .. 968
ETBM0580 NUM-LONG-BUFFER ........... 0
ETBM0580 Long buffers active ....... 0
ETBM0580 Long buffers active HWM ... 0
ETBM0580 NUM-SHORT-BUFFER .......... 0
ETBM0580 Short buffers active ...... 1219
ETBM0580 Short buffers active HWM .. 1928
ETBM0582 Function completed
```

ETBEND

Processing stops immediately. Current calls to the EntireX Broker are not allowed to finish.

ETBSTOP

Alias of `ETBEND`.

FLUSH

Flush all trace data kept in internal trace buffers to stderr (`SYSOUT`). The broker-specific attribute `TRMODE=WRAP` is required.

PSTORE TRACE

Modifies the trace level for the Adabas persistent store (Adabas-specific attribute `TRACE-LEVEL`).

Example

To set a trace level 2 for the Adabas persistent store

- Enter command:

```
/INTR tsn,PSTORE TRACE=2
```

See `TRACE-LEVEL` under *Adabas-specific Broker Attributes*.

SHUTDOWN class,server,service

Shuts down the specified service immediately and stops all servers that have registered this service.

Example

To shut down service `CLASS=RPC, SERVER=SRV1, SERVICE=CALLNAT`

- Enter command:

```
/INTR tsn,SHUTDOWN RPC,SRV1,CALLNAT
```

TRACE

Modifies the setting of the broker-specific attribute `TRACE-LEVEL`.

Sample Commands

To modify the trace level

- Enter command, for example:

```
/INTR tsn,TRACE=0
/INTR tsn,TRACE=1
/INTR tsn,TRACE=4
```

See `TRACE-LEVEL` under *Broker-specific Broker Attributes*.

TRAP-ERROR

Modifies the setting of the broker-specific attribute `TRAP-ERROR`.

Sample Command

To modify the setting for `TRAP-ERROR`

- Enter command:

```
/INTR tsn,TRAP-ERROR=nnnn
```

  where *nnnn* is the four-digit API error number that triggers the trace handler.

See `TRAP-ERROR` under *Broker-specific Broker Attributes*.

Participant-specific Commands

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. The following participant-specific commands are available:

- CANCEL parameter
- USERLIST
- USERS parameter

CANCEL parameter

Operator command `CANCEL` is used to delete participants from EntireX Broker. The following parameters are supported:

Parameter	Description
`[USER=]`*`user_id`*	Cancel all participants with the specified *`user_id`*. Non-persistent resources will be freed by the timeout manager. Prefix "`USER=`" is the default value and may be omitted.
`SEQNO=`*`seqno`*	Cancel the participant with the sequence number *`seqno`*. Non-persistent resources will be freed by the timeout manager. Operator commands `USERLIST` and `USERS` display sequence numbers of all selected participants.

Sample Commands

To cancel all participant entries of user "DOE"

- Enter command:

```
/INTR tsn,CANCEL DOE
```

Or:

```
/INTR tsn,CANCEL USER=DOE
```

To cancel participant with sequence number "11"

- Enter command:

```
/INTR tsn,CANCEL SEQNO=11
```

USERLIST

Operator command `USERLIST` displays a list of selected participant entries. The following parameters are supported:

Parameter	Description
none | *	Display all participants.
user_id	Display all participants with user ID *user_id*. Wildcard characters are supported.

Sample Commands

To display all participants

- Enter command:

```
/INTR tsn,USERLIST
```

Or:

```
/INTR tsn,USERLIST *
```

To display all participants with user ID "DOE"

- Enter command:

```
/INTR tsn,USERLIST DOE
```

This produces the following output. See *Description of USERLIST Output Columns* below.

```
ETBM0720 Operator typed in: USERLIST DOE
ETBM0687 Participants:
ETBM0687 USER-ID ------------------------ C S E CHR SEQNO
ETBM0687 DOE                              N Y Y ASC 1
ETBM0582 Function completed
```

To display all participants with user ID starting with uppercase "D"

- Enter command:

```
/INTR tsn,USERLIST D*
```

This produces the following output. See *Description of USERLIST Output Columns* below.

```
ETBM0720 Operator typed in: USERLIST D*
ETBM0687 Participants:
ETBM0687 USER-ID ------------------------ C S E CHR SEQNO
ETBM0687 DOE                              N Y Y ASC 1
ETBM0687 DOE1                             N Y Y EBC 2
ETBM0687 DOE2                             N Y Y EBC 3
ETBM0687 DOE3                             N Y Y EBC 4
ETBM0582 Function completed
```

To display all participants with 4-character user ID, starting with uppercase "D" and with uppercase "E" as third character

- Enter command:

```
/INTR tsn,USERLIST D?E?
```

This produces the following output. See *Description of USERLIST Output Columns* below.

```
ETBM0720 Operator typed in: USERLIST D?E?
ETBM0687 Participants:
ETBM0687 USER-ID ------------------------ C S E CHR SEQNO
ETBM0687 DOE1                             N Y Y EBC 2
ETBM0687 DOE2                             N Y Y EBC 3
ETBM0687 DOE3                             N Y Y EBC 4
ETBM0582 Function completed
```

Description of USERLIST Output Columns

Keyword	Description
USER-ID	User ID (32 bytes, case-sensitive). See USER-ID under *Broker ACI Fields*.
C	Client. Y Participant is a client, otherwise "N".
S	Server. Y Participant is a server, otherwise "N".
E	Big endian. Y Participant is on a big-endian machine. N Participant is on a little-endian machine.

Keyword	Description
CHR	Character set. ASC Participant is an ASCII user. EBC Participant is an EBCDIC user.
SEQNO	Sequence number of participant. Can be used for operator command CANCEL *parameter*.

USERS parameter

Operator command USERS displays selected user data of participant entries. The following parameters are supported:

Parameter	Description
none | *	Display all participants.
user_id	Display all participants with user ID *user_id*. Wildcard characters are supported.

Sample Commands

To display all participants

- Enter command:

```
/INTR tsn,USERS
```

Or:

```
/INTR tsn,USERS *
```

To display all participants with user ID "DOE"

- Enter command:

```
/INTR tsn,USERS DOE
```

This produces the following output. See *Description of USERS Output Columns* below.

```
ETBM0720 Operator typed in: USERS DOE
ETBM0687 Participants:
ETBM0687 USER-ID: DOE
ETBM0687 CLIENT: N SERVER:
ETBM0687 SEQNO: 6 BIG ENDIAN: Y CHARSET: ASCII PUID:
ETBM0687 202073756E6578322D2D30303030324646462D2D3030303030303031
ETBM0687 TOKEN:
ETBM0582 Function completed
```

Description of USERS Output Columns

Keyword	Description
USER-ID	User ID (32 bytes, case-sensitive). See USER-ID under *Broker ACI Fields*.
CLIENT	Y Participant is a client, otherwise "N".
SERVER	Y Participant is a server, otherwise "N".
BIG ENDIAN	Y Participant is on a big-endian machine. N Participant is on a little-endian machine.
CHARSET	ASC Participant is an ASCII user. EBC Participant is an EBCDIC user.
PUID	Internal unique ID of participant. Hexadecimal 28-byte value in printable format.
TOKEN	Optionally identifies the participant. See TOKEN under *Broker ACI Fields*.

Security-specific Commands

DSECSTAT

Displays the number of successful and failed Security authentications and Security authorizations.

Sample Output

```
ETBM0720 Operator typed in: DSECSTAT
ETBM0579 Security Authentications - successful: 20 failed: 0
ETBM0579 Security Authorizations  - successful: 0 failed: 0
```

RESET userid

Resets the Security context for the specified user ID.

Sample Output

```
ETBM0720 Operator typed in: RESET EXXBATCH
ETBM0578 Reset ACEE for SAF-ID EXXBATCH : 20 instances found
```

SECURITY TRACE

Modifies the trace level for the EntireX Security (security-specific attribute `TRACE-LEVEL`). Broker-specific attribute `SECURITY=YES` must be set.

Example

To set a trace level 2 for EntireX Security

- Enter command:

```
/INTR tsn,SECURITY TRACE=2
```

See `TRACE-LEVEL` under *Security-specific Broker Attributes*.

Transport-specific Commands

Transport-specific commands are available for Adabas/Entire Net-Work communicators, SSL communicators and TCP communicators; the `COM` command can be used for all communicators. The following command syntax applies:

```
                  { COM  }   { STATUS      }
                  { NET  }   { SUSPEND     }
/INTR tsn,        { SSL  }   { RESUME      }
                  { Snn  }   { STOP        }
                  { TCP  }   { START       }
                  { Tnn  }   { TRACE={0-8} }
```

COM parameter

This command is executed by all configured transport communicators. The following parameters are supported:

Parameter	Description
`STATUS`	Displays the current status of the transport communicator.
`SUSPEND`	Used to suspend the transport communicator. The transport communicator is halted but will not shut down. User requests receive response code 148.
`RESUME`	Resume a suspended transport communicator. If the communicator was not suspended before, an error message will be displayed.
`STOP`	Stop an active or suspended transport communicator. The transport communicator will shut down. All transport-specific resources will be freed. User requests receive response code 148.
`START`	Start a transport communicator that was previously stopped. If the communicator was not stopped before, an error message will be displayed.

`TRACE`	Sets the trace level for the transport method. If the global trace level (see `TRACE`) is set with command
	`/INTR tsn,TRACE=n`
	this applies to *all* transport methods. This command will also override any existing transport-specific settings. If you subsequently enter command
	`/INTR tsn, TCP TRACE=n`
	only the trace level for TCP/IP transport is modified.
	Note: With commands `TCP Tnn`, and `SSL` and `Snn`, the trace level is set for *all* TCP and SSL communicators respectively. Setting a trace level for a single TCP or SSL instance is not supported. For example: although it is possible to submit the command
	`/INTR tsn,T01 TRACE=1`
	this command sets the trace level for all TCP communicators.

Sample Output

```
ETBM0720 Operator typed in: COM STATUS
ETBW0718 TCP Communicator 0 currently active
ETBW0718 TCP Communicator 1 currently active
ETBW0718 SSL Communicator 0 currently suspended
ETBW0718 NET Communicator 0 currently suspended
XC00039I 00113 Total number of commands = 17
XC00057I 00113 Operator entry active
ETBM0720 Operator typed in: COM SUSPEND
ETBM0721 TCP Communicator 0 suspended
ETBM0721 TCP Communicator 1 suspended
ETBM0721 SSL Communicator 0 suspended
ETBM0721 NET Communicator 0 suspended
```

NET parameter

This command is executed by X-COM, the Adabas/Entire Net-Work communicator. See command `COM` above for a list of supported parameters.

Sample Output

```
ETBM0720 Operator typed in: NET STATUS
ETBW0718 NET Communicator 0 currently active
XC00039I 00113 Total number of commands = 17
XC00057I 00113 Operator entry active
```

SSL parameter

This command is executed by all SSL communicators. See command `COM` above for a list of supported parameters.

Sample Output

```
ETBM0720 Operator typed in: SSL STATUS
ETBW0718 SSL Communicator 0 currently active
```

To manipulate a specific communicator instance (max. five instances can be started), use the command `S00`, `S01`, `S02`, `S03` or `S04` for the respective SSL instance.

TCP parameter

This command is executed by TCP communicators. See command `COM` above for a list of supported parameters.

Sample Output

```
ETBM0720 Operator typed in: TCP STATUS
ETBW0718 TCP Communicator 0 currently active
ETBW0718 TCP Communicator 1 currently active
```

```
ETBM0720 Operator typed in: TCP RESUME
ETBM0721 TCP Communicator 0 resumed
ETBM0721 TCP Communicator 1 resumed
```

To manipulate a specific communicator instance (max. five instances can be started), use the command `T00`, `T01`, `T02`, `T03` or `T04` for the respective TCP instance.

Sample Output

```
ETBM0720 Operator typed in: T00 STATUS
ETBW0718 TCP Communicator 0 currently active
```

```
ETBM0720 Operator typed in: T01 STATUS
ETBW0718 TCP Communicator 1 currently active
```

Sample Transport Commands

To display status of all transport communicators

- Enter command:

```
/INTR tsn,COM STATUS
```

To suspend first TCP communicator

- Enter command:

```
/INTR tsn,T00 SUSPEND
```

To stop all SSL transport communicators

- Enter command:

```
/INTR tsn,SSL STOP
```

XCOM-specific Commands

Note: All operator commands beginning with "X" belong to X-COM, the Adabas/Entire Net-Work communicator. The following commands operate only on the Adabas transport mechanism: `XCQES`, `XHALT`, `XPARM`, `XSTART`, `XSTAT` and `XUSER`. These commands have no effect on functions not related to the Adabas transport mechanism.

`XEND` and `XSTOP` function independently of the transport mechanism. (They stop the Broker's processing immediately, whereby existing calls to the EntireX Broker are not allowed to finish.)

XABS

Displays the total size, the number of bytes in use, the number of free bytes and the largest free windows in the Adabas attached buffer pool on the console.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

Sample Output

```
ETBM0720 Operator typed in: XABS
XC00090I 10113 Attached buffer usage
XC00090I 10113   38912000 bytes total = 9500 NABS
XC00090I 10113          0 bytes used
XC00090I 10113          0 bytes used HWM
XC00090I 10113   38912000 bytes free
XC00090I 10113   38912000 bytes current largest free windows
XC00090I 10113   38912000 bytes minimum of all largest free windows
```

XCQES

Displays the current number, and the highest number, of Adabas command queue elements to the console.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

Sample Output

```
ETBM0720 Operator typed in: XCQES
XC00030I 00113 Number of active CQEs = 0
XC00031I 00113 Highest number of active CQEs = 1
```

XEND

Alias of ETBEND.

XHALT

New calls to the EntireX Broker are temporarily rejected. Processing is resumed by issuing the XSTART operator command. XHALT is an alias for command NET SUSPEND.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

Sample Output

```
ETBM0720 Operator typed in: XHALT
ETBM0721 NET Communicator 0 suspended
```

XPARM

Displays the values of Adabas SVC, database ID, number of CQEs, number of attached buffers, and the application name for the Adabas transport to the console.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

Sample Output

```
ETBM0720 Operator typed in: XPARM
XC00032I 00113 Parameters for this session:
XC00033I 00113 SVC = 249
XC00034I 00113 NODE = 00113
XC00035I 00113 NCQE = 00100
XC00036I 00113 NABS = 10000
XC00037I 00113 User application = ETBNUC
```

XSTART

Processing of new calls to the EntireX Broker, interrupted with the `XHALT` command, is resumed. `XSTART` is an alias of command `NET RESUME`.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

Sample Output

```
ETBM0720 Operator typed in: XSTART
ETBM0721 NET Communicator 0 resumed
```

XSTAT

Displays the EntireX Broker statistics as console messages.

Note: This command operates on the Adabas transport mechanism only. It has no effect on functions not related to the Adabas transport mechanism.

XSTOP

Alias of `ETBEND`.

XUSER

Displays the current number, as well as the highest number, of users actively issuing commands using the Adabas transport mechanism to the console.

Note: The number of users displayed with this operator command will not represent all of the Broker clients and servers but only the subset of users issuing commands using the Adabas transport mechanism. Command and Information Services provides comprehensive information about all Broker clients and servers.

11 Administering the RPC Server for BS2000

- Customizing the RPC Server 136
- Configuring the RPC Server 137
- Locating and Calling the Target Server 144
- Starting the RPC Server 145
- Stopping the RPC Server 145
- Activating Tracing for the RPC Server 153

The EntireX RPC Server for BS2000 allows standard RPC clients to communicate with RPC servers on the operating system BS2000. It supports the programming languages COBOL and C.

For more information on this component see the separate *RPC Server for BS2000* documentation.

Customizing the RPC Server

The following elements are used for setting up the RPC Server for BS2000:

- Common Runtime Environment (CRTE)
- Configuration File
- Start Procedure

Common Runtime Environment (CRTE)

When the RPC Server for BS2000 calls COBOL or C server programs, the BS2000 Common Runtime Environment (CRTE) is loaded dynamically into the corresponding address space of the worker task.

There is no need to bind the CRTE statically to the called server object modules. If this is needed for any reason, the CRTE must be linked as a subsystem. All entries must be hidden to prevent duplicates. Linking the CRTE statically will occupy resources and slow down the load time of the server object modules.

The CRTE is not delivered with this package. For a detailed description, see the *CRTE (BS2000) User's Guide*.

Configuration File

The name of the delivered example configuration file is `RPC_CONFIG`. The configuration file contains the configuration for the RPC Server for BS2000. The following settings are important:

- connection information such as broker ID, server address (class, name, service)
- scalability parameters
- trace settings
- etc.

For more information see ***Configuring the RPC Server***.

Start Procedure

The name of the start S-procedure for the RPC Server for BS2000 is "START-RPC-SERVER". The start procedure contains the following:

- the location of the Common Runtime Environment (CRTE)
- the target server library name of the called COBOL or C server
- the configuration file used; see *Configuration File*
- etc.

Configuring the RPC Server

The following rules apply:

- In the configuration file:
  - Comments must be on a separate line.
  - Comment lines can begin with '*', '/' and ';'.
  - Empty lines are ignored.
  - Headings in square brackets [<topic>] are ignored.
  - Keywords are case-insensitive.
- Underscored letters in a parameter indicate the minimum number of letters that can be used for an abbreviated command.

  For example, in `brokerid=localhost`, `brok` is the minimum number of letters that can be used as an abbreviation, that is, the commands/parameters `broker=localhost` and `brok=localhost` are equivalents.

Parameter	Default	Values	Req/ Opt
`brokerid`	`localhost`	Broker ID used by the server. See *Using the Broker ID in Applications* in the RPC Programming documentation. Example: `brokerid=myhost.com:1971`	R
`class`	`RPC`	Server class part of the server address used by the server. The server address must be defined as a service in the broker attribute file (see *Service-specific Attributes*). Case-sensitive, up to 32 characters. Corresponds to `CLASS` attribute of the broker attribute file.	R

Parameter	Default	Values	Req/ Opt
		Example: `class=MyRPC`	
`codepage`	no codepage transferred	The codepage tells the broker the encoding of the data. The application must ensure the encoding of the data matches the codepage. The RPC server itself does not convert your application data. The application's data is shipped and received as given. Often, the codepage must also match the encoding used in the RPC server environment for file and terminal IO, otherwise unpredictable results may occur. By default, no codepage is transferred to the broker. It is assumed the broker's locale string defaults match. See *Locale String Mapping* If they do not match, provide the codepage here. Example: `codepage=EDF041` Enable character conversion in the broker by setting the service-specific attribute `CONVERSION` to "`SAGTRPC`". See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.	R

`compresslevel`	`N`	Enforce compression when data is transferred between broker and server. See *Data Compression in EntireX Broker* in the platform-independent Administration documentation.
		
		`compresslevel=0|1|2|3|4|5|6|7|8|9|Y|N`
		
		`0-9` 0=no compression
		9=max. compression
		`N` no compression
		`Y` compression level 6
		
		Example:
		`compresslevel=6`
			O

`deployment`	`NO`	Activates the deployment service, see *Deployment Service*. Required to use the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the Designer documentation.	O

Parameter	Default	Values	Req/ Opt
		`YES` Activates the deployment service. The RPC server registers the deployment service in the broker. `NO` The deployment service is deactivated. The RPC server does not register the deployment service in the broker. Example: `deployment=yes`	
`init_exit`		Initialization exit. The RPC Server for BS2000 provides user exits that allow you to plug in code during initialization and termination of RPC worker tasks. This parameter specifies the name of an executable module that is loaded and executed during initialization of each worker task. See also `term_exit`. Example: `init_exit=myExit`	O

`extractor`	`NO`	The extractor service is a prerequisite for remote extractions. See *Extractor Service*.
		
		`extractor=YES|NO`
		
		Example:
		`extractor=yes`
			O

`logon`	`YES`	Execute broker functions `LOGON/LOGOFF` in worker threads. Must match the setting of the broker attribute `AUTOLOGON`. Reliable RPC requires logon set to `YES`. See *Reliable RPC*. `NO` No logon/logoff functions are executed. `YES` Logon/logoff functions are executed. Example: `logon=no`	O
`marshalling`	`COBOL`	The RPC Server for BS2000 can be configured to support either COBOL or C. See also ***Locating and Calling the Target Server***. `marshalling=(LANGUAGE=COBOL|C)` `COBOL` The RPC Server for BS2000 supports COBOL. The COBOL servers are called directly without a server interface object. The COBOL server modules may be compiled as OM or	O

Parameter	Default	Values	Req/ Opt
		LLM modules. So-called server mapping files are used to call the COBOL server correctly if one is available. See *Usage of Server Mapping Files* in the RPC Server for BS2000 documentation. `C` The RPC Server for BS2000 supports C. The modules are called using a server interface object built with the C Wrapper.	
`password`	no default	The password for secured access to the broker. Example: `password=MyPwd`	O
`restartcycles`	15	Number of restart attempts if the broker is not available. This can be used to keep the RPC Server for BS2000 running while the broker is down for a short time. A restart cycle will be repeated every 60 seconds. When the number of specified cycles is reached and a connection to the broker is not possible, the RPC Server for BS2000 stops. Example: `restartcycles=30` The server waits up to 30 minutes before it terminates due to a missing broker connection.	O
`servername`	`SRV1`	Server name part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes*. Case-sensitive, up to 32 characters. Corresponds to `SERVER` of the broker attribute file. Example: `servername=mySrv`	R
`service`	`CALLNAT`	Service part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes*. Case-sensitive, up to 32 characters. Corresponds to `SERVICE` attribute of the broker attribute file. Example: `service=MYSERVICE`	R
`svm`	`PREFERRED`	Usage of server-side mapping files (Designer files with extension .svm) at runtime. See *Server-side Mapping Files in the RPC Server*. If you use server-side mapping	O

Parameter	Default	Values	Req/ Opt

		files, the server-side mapping container must be installed and configured. See *Step 1: Define a Server-side Mapping Container* in the BS2000 Installation documentation.
		
		Server mapping files with extension .svm are no longer supported at design time by the Designer. You can still use them at runtime in a server-side mapping container. All special COBOL syntax and features supported by server mapping files with extension .svm are also covered by server mapping files with extension .cvm. See *When is a Server Mapping File Required?*
		We recommend migrating .svm files to .cvm files. See *Migrating Server Mapping Files* under *Server Mapping Files for COBOL* in the Designer documentation.
		
		`SVM=PREFERRED|NO`
		
		`PREFERRED` This setting is to support COBOL server programs that do not have server-side mapping, as well as COBOL server programs built with a server-side mapping file.
		`NO` Server-side mapping files are not used.
		
		Example:
		`SVM=NO`
		
		See also *Usage of Server Mapping Files*.	

`term_exit`		Termination exit. The RPC Server for BS2000 provides user exits that allow you to plug in code during initialization and termination of RPC worker tasks. This parameter specifies the name of an executable module that is loaded and executed during termination of each worker task. See also `init_exit`. Example: `term_exit=myExit`	O
`timeout`	60	Timeout in seconds, used by the server to wait for broker requests. See `WAIT` under *Broker ACI Fields* for more information. Also influences `restartcycles` and worker model *`DYNAMIC`*. Example: `timeout=300`	O

Parameter	Default	Values	Req/ Opt
tracedestination	ERXTrace.*nnn*.log	Trace output is written to SYSOUT. See also ***Activating Tracing for the RPC Server***.	O
tracelevel	None	Trace level for the server. See also ***Activating Tracing for the RPC Server***. tracelevel=None|Standard|Advanced|Support None — No trace output. Standard — For minimal trace output. Advanced — For detailed trace output. Support — This trace level is for support diagnostics. Use only when requested by Software AG Support. Example: tracelevel=standard	O
userid	ERX-SRV	The user ID for access to the broker. The default ERX-SRV will be used if this parameter is omitted or specified without a value: "userid=". Example: userid=MyUid	O
workermodel	SCALE,1,3,slowshrink	The RPC Server for BS2000 can be configured to ■ use a *DYNAMIC* worker model, which adjusts the number of worker threads to the current number of client requests: workermodel=(SCALE,*from*,*thru* [,slowshrink|fastshrink]) ■ use a *FIXED* number of worker threads: workermodel=(FIXED,*number*) FIXED — A fixed *number* of worker threads is used by the RPC Server for BS2000. SCALE — The number of worker threads is adjusted to the current number of client requests. With the *from* value, the minimum number of active worker threads can be set. This allows you to define a certain number of threads - not used by the currently executing RPC	O

Parameter	Default	Values	Req/ Opt
		request - to wait for new RPC client requests to process. In this way the RPC server is ready to handle many RPC client requests arriving at the same time. The *thru* value restricts the maximum number of all worker threads concurrently. ■ `slowshrink` Default. The RPC server stops all worker threads not used in the time specified by the `timeout` parameter, except for the number of workers specified as minimum value. ■ `fastshrink` The RPC server stops worker threads immediately as soon as it has finished its conversation, except for the number of workers specified as minimum value. Example: `workermodel=(SCALE,2,5)`	

Locating and Calling the Target Server

Target server programs are loaded dynamically, using the BS2000 BLSLIB chain. The target server library name needs to be set up as `PROGRAM-LIB` in the parameter declaration section of the `START-RPC-SERVER` S-procedure, see ***Start Procedure***. Different mechanisms are used depending on the language:

- COBOL
- C

COBOL

The COBOL object module name for the RPC server called is taken from the server mapping if one is available. See *Usage of Server Mapping Files* for an introduction. If no server mapping is used, the IDL program name is used as the COBOL object module name of the RPC server and the IDL library name is ignored.

See also *Scenario I: Calling an Existing COBOL Server* or *Scenario II: Writing a New COBOL Server*.

To use the RPC Server for BS2000 with COBOL

1 Make sure that all target server programs called as RPC servers

- are COBOL object modules
- use COBOL calling conventions

2 Configure the parameter `marshalling` for COBOL, for example:

```
marshalling=COBOL
```

C

To use the RPC Server for BS2000 with C

1 Make sure that all target server programs called as RPC servers

- are C object modules
- use C calling conventions

2 Configure the parameter `marshalling` for C, for example:

```
marshalling=C
```

See *Scenario III: Writing a New C Server* in the RPC Server for BS2000 documentation.

Starting the RPC Server

To start the RPC Server for BS2000

- Use the following SDF command:

```
/ENTER-PROCEDURE *LIB(LIB=EXP103.JOBS,ELE=START-RPC-SERVER), -
/JOB-NAME=RPCMAIN,LOG=*NO
```

Stopping the RPC Server

To stop the RPC Server for BS2000 from a privileged user ID

- Enter the command:

```
/INFORM-PROGRAM MSG='STOP',JOB-IDENTIFICATION=*TSN(TSN=tsn)
```

where *tsn* is the task number associated with the RPC Server for BS2000 main task (in the example above the TSN of `RPCMAIN`)

All other tasks that were created as a result of starting the RPC Server for BS2000 will be stopped automatically.

To stop the RPC Server for BS2000 from an operator console

- Enter the command:

```
/INTR tsn,STOP
```

where *tsn* is the task number associated with the RPC Server for BS2000 main task (in the example above the TSN of `RPCMAIN`)

All other tasks that were created as a result of starting the RPC Server for BS2000 will be stopped automatically.

To stop the RPC Server for BS2000 from a non-privileged user ID

- Use S-procedure `STOP-RPC-SERVER` in `EXP103.JOBS`.

Startup Parameter	Description	Default

`BROKER-ID`	Depending on the communication method, the broker ID can be specified in two different formats:
	■ **TCP Transport Method**
	`ip:port:TCP`
	where *ip* is the address or DNS host name,
	port is the port number that EntireX Broker is listening on, and
	`TCP` is the protocol name
	■ **NET Transport Method**
	`ETBnnn:SVCmmm:NET`
	where *nnn* is the ID under which EntireX Broker is connected to the Adabas ID table,
	mmm is the SVC number under which the Adabas ID table can be accessed, and
	`NET` is the protocol name
		none

`CLASS`	The class name under which the RPC server is registered at the EntireX Broker.	`RPC`
`SERVER`	The server name under which the RPC server is registered at the EntireX Broker.	`SRV1`
`SERVICE`	The service name under which the RPC server is registered at the EntireX Broker.	`CALLNAT`
`USERID`	If EntireX Broker is running with EntireX Security, a user ID needs to be supplied.	none
`PASSWORD`	If EntireX Broker is running with EntireX Security, a password needs to be supplied.	none
`EXX-JOBS`	EntireX Broker jobs library.	`EXX103.JOBS`
`EXX-LIB`	EntireX Broker module library.	`EXX103.LIB`
`WAL-MOD`	WAL module library.	`WAL842.MOD`

Set the broker ID in the `PARAMETER-DECLARATION` section and enter following command:

```
/CALL-PROCEDURE (EXP103.JOBS, STOP-RPC-SERVER)
```

Activating Tracing for the RPC Server

To switch on tracing for the RPC server

- Set the parameter `TRACELEVEL` in S-element `RPC-CONFIG` in `EXP103.JOBS`.

  To evaluate the RPC server return codes, see *EntireX RPC Server Return Codes* in the Error Messages and Codes documentation.

12 Tracing EntireX Components under BS2000

- Tracing EntireX Broker 150
- Tracing Broker Stubs 151
- Activating Tracing for the RPC Server 153
- Tracing Broker Security Server 153

Tracing EntireX Broker

- Switching on Tracing
- Switching off Tracing
- Deferred Tracing
- Dynamically Switching On or Off the EntireX Broker Trace

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.

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.
- **Operator Command**
  Issue an operator command. See TRACE.

  See also *Deferred Tracing*.

Tracing Broker Stubs

- Scope
- Setting Trace Options

- Switching off Tracing

Scope

The following table describes the possible values for the broker stub trace:

Trace Level		Description
0	NONE	No tracing. Switch tracing off.
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 information from the transports.
3	SUPPORT	This is full tracing through the stub, including detailed traces of control blocks, message information, etc.

Setting Trace Options

Trace options can be supplied to broker stubs for NET communication as well TCP/SSL communication via SDF job control statements or JV job variables:

Transport	SDF Job Control	JV Control	Output
TCP/SSL	/ETB-STUBLOG = '3'	CREATE-JV ETB-STUBLOG MODIFY-JV ETB-STUBLOG,SET=C'3'	<tsn>.ETB
NET	/EXA-LOG = '3'	CREATE-JV EXA-LOG MODIFY-JV EXA-LOG,SET=C'3'	<tsn>.EXA

Switching off Tracing

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

- Set the SDF job control option to '0' or delete it.
- Set the JV job variable to '0' or delete it, for example

```
DELETE-JV ETB-STUBLOG
```

Activating Tracing for the RPC Server

To switch on tracing for the RPC server

- Set the parameter `TRACELEVEL` in S-element `RPC-CONFIG` in `EXP103.JOBS`.

  To evaluate the RPC server return codes, see *EntireX RPC Server Return Codes* in the Error Messages and Codes documentation.

Tracing Broker Security Server

The Broker Security Server comes with a trace facility that can be used to track the IDs of users logging on to EntireX Broker. It also produces some diagnostic messages that are helpful for problem analysis. By default, no tracing is performed.

To switch on tracing for Broker Security Server

- Set up an `SDF` variable in the server's job control.

```
TRACE='ON'
```

To switch off tracing for Broker Security Server

- Set the following in the server's job control.

```
TRACE='OFF'
```

See *EntireX Broker Security Server for BS2000*.

13 Broker Shutdown Statistics

- Shutdown Statistics Output 156
- Table of Shutdown Statistics 156

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.

14 Command Logging in EntireX

- Introduction to Command Logging 162
- ACI-driven Command Logging 164
- Dual Command Log Files 164

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

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

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

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

Introduction to Command Logging

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

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

Overview

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

The following Broker attributes are involved in command logging:

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

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

Command Log Files

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

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

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.

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.

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`.

15 Accounting in EntireX Broker

- EntireX Accounting Data Fields 166
- Example Uses of Accounting Data 169

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 10.3.0.00.
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.

Example Uses of Accounting Data

- Chargeback
- Trend Analysis
- Tuning for Application Performance

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, the owner 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 Linux. 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.