Setting up Broker Instances

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

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

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* under *Broker 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* under *Broker Attributes*.

2. Ensure that communication with the broker is possible by running the installation verification programs (bcoc, bcos) using transport type NET.

Starting and Stopping the Broker

Starting the Broker

To start the broker

- Enter the following SDF command:

```
/ENTER-PROCEDURE *LIB(LIB=EXX811.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 EXX811.JOBS

Startup Parameter	Description	Default
BROKER-ID	Depending on the communication method, the BrokerId 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** ETB*nnn*:SVC*mmm*: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.	EXX811.LIB
EXX-JOBS	EntireX Broker jobs library.	EXX811.JOBS
WAL-MOD	WAL module library.	WAL826.MOD

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

```
/CALL-PROCEDURE (EXX811.JOBS, STOP-BROKER)
```

Tracing EntireX Broker

This section covers the following topics:

- Broker TRACE-LEVEL Attribute
- Attribute File Trace Setting
- Deferred Tracing

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. You can also modify the different `TRACE-LEVEL` values while Broker is running, without having to restart the Broker kernel for the change to take effect.

Individual Settings	Specified in Attribute File Section
Master trace level	`DEFAULTS=BROKER`
Persistent Store trace level	`DEFAULTS=ADABAS`
Conversion trace level	Trace option of the `CONVERSION` parameter that can be defined in `DEFAULTS=SERVICE | TOPIC`
Security trace level	`DEFAULTS=SECURITY`
Transport trace level	`DEFAULTS=NET | TCP | SSL`

These individual `TRACE-LEVEL` values determine the level of tracing within each subcomponent. If not specified, the master `TRACE-LEVEL` is used.

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.
8	All of Trace Level 4, plus Adabas Persistent Store Adabas control blocks.

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".

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.