Administering Broker Stubs

This chapter covers the following topics:

- Available Stubs
- BROKER
- CICSETB
- COMETB
- IDMSETB
- MPPETB
- NATETB23
- Transport Methods for Broker Stubs
- Tracing for Broker Stubs
- SVC Number for Broker Communication
- SAGTOKEN Utility
- Support of Clustering in a High Availability Scenario
- Considerations for Users without Adabas

Available Stubs

This table lists all broker stubs available under z/OS are to be used with the programming languages Assembler | C | COBOL | Natural | PL/I.

Your selection of a specific broker stub depends on the following:

- the environment (TP monitor, TSO, Batch, Natural)
- the transport method (NET, TCP, or SSL)
- the availability of administration features such as trace, and of function features such as compression

Note:
Use of the 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 z/OS. In order to use the transport method NET for messages involving more than 32 KB, you must install Adabas cross-memory services (see *Considerations for Users without Adabas*). If you have not yet installed Adabas cross-memory services, you can instead use TCP/IP to transport more than 32 KB of data.

Environment	Transport			Transport Timeout	Trace	Compression	Stub Module
	NET	TCP	SSL				
Batch, TSO	Yes	Yes	Yes	Yes	Yes	Yes	*BROKER*
CICS	Yes	Yes	Yes	Yes	Yes	Yes	*CICSETB*
IDMS/DC	No	Yes	No	No	No	Yes	*IDMSETB*
IMS (BMP)	Yes	Yes	Yes	Yes	Yes	Yes	*BROKER*
IMS (MPP)	Yes	Yes	Yes	Yes	Yes	Yes	*MPPETB*
Com-plete	Yes	Yes	Yes	Yes	Yes	Yes	*COMETB*
Natural	Yes	Yes	Yes	Yes	Yes	Yes	*NATETB23*

BROKER

Scope

See *Available Stubs* above for overview of functionality and considerations.

Prerequisites and Installation Notes

This stub can be used in a multithreading (subtasking) environment, provided ADAUSER is not linked to the application.

Notes:

1. It is recommended to load BROKER dynamically within the application program and not to link BROKER with any Adabas link routine. BROKER will attempt to load ADALNKR dynamically.
2. However, if BROKER has to be statically linked, see the subsection *Linkage* below.
3. BROKER is the recommended stub for any batch environment. It provides all available stub features.

Default Transport Method

BROKER checks DD assignments in order to determine the default transport method. The JCL statement `//EXATCP DD DUMMY` is used to define TCP as default transport. `//EXANET DD DUMMY` is used to define NET as default transport. NET is already the default value.

Linkage

Choose the method most appropriate for your application:

- Method 1: Reentrant (Thread-Safe)
- Method 2: Non-reentrant

Method 1: Reentrant (Thread-Safe)

Link your application to module BROKER from the EntireX load library (EXX970.LOAD).

Linkage statements:

```
INCLUDE userlib(mainpgm) Main Program
INCLUDE exxlib(BROKER)  Broker stub
ENTRY mainpgm
NAME ...
```

The SVC number may be specified as part of the Broker ID, for example:

```
ETB220:SVC237:NET
```

Method 2: Non-reentrant

Link your application to module BROKER from the EntireX load library (EXX970.LOAD) and the module ADAUSER from the Adabas load library.

Linkage statements:

```
INCLUDE userlib(mainpgm) Main Program
INCLUDE exxlib(BROKER)   Broker stub
INCLUDE wallib(ADAUSER)  Adabas batch/TSO front end
ENTRY mainpgm
NAME ...
```

Notes:
Linking with ADAUSER provides the ability to specify the required SVC in the Adabas DDCARD parameter of the application job, as shown below:

```
//J020S1   EXEC PGM=PROGRAM
//STEPLIB  DD DISP=SHR,DSN=EXX970.LOAD
//         DD DISP=SHR,DSN=WAL826.LOAD
//ETBPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DDCARD   DD *
ADARUN MODE=MULTI,PROGRAM=USER,SVC=237
/*
```

When linking the stub for use in IMS (BMP), please substitute the appropriate Adabas link module for this environment.

Whichever linkage method you select, you *must* ensure that library EXX970.LOAD is in the steplib of the application and that library WAL826.LOAD is in the steplib when using Adabas [NET] transport.

When using any of the batch stubs, you must ensure the Adabas link routine (ADALNK/ADALNKR) does not contain any exits which assume the length of the Adabas data area UB is extended beyond its default value. Contact Software AG support if you are unsure about this.

CICSETB

Scope

See *Available Stubs* above for overview of functionality and considerations.

Prerequisites and Installation Notes

- EntireX z/OS CICS® RPC Server uses CICSETB as default.
- It is recommended to load CICSETB dynamically within the application program.
- CICSETB will attempt to load ADACICS dynamically, using `EXEC CICS LINK PROGRAM`.
- CICSETB is the recommended stub for any CICS environment.
- CICSETB must be available in the CICS RPL search chain if called dynamically. Alter the CICS procedure or job, adding the EXX load library to both the STEPLIB and DFHRPL library concatenations.
- See the CICS-related entries for EXAGLUE, EXAMEM and CICSETB2 in job EXXCICS in the EXX jobs library.
- CICSETB can be used with or without a CICS TWA (transaction work area).
    - **With TWA**
      At least 24 bytes of transaction work area must be defined in your CICS transaction if you choose to specify a TWA.
    - **Without TWA**
      Prerequisite is building an Adabas CICS interface, specifying `PARMTYP=ALL` for either the ADAGSET or LGBLSET MACRO (depending upon the Adabas version). There are no application changes required for using CICSETB without TWA.
    - If the name of your Adabas CICS link routine is different from "ADACICS", please use zap EXX970.ZAPS(EXX0007) to set the site-specific name of your Adabas CICS link routine. The zap will change the name to "ADABAS".

Linkage

It is also possible to link your application to member CICSETB from the EntireX load library (EXX970.LOAD).

Linkage statements for COBOL applications:

```
INCLUDE cicslib(DFHECI)   CICS Prolog Module
INCLUDE userlib(mainpgm)  Main Program
INCLUDE cicslib(DFHELII)  CICS Module
INCLUDE exxlib(CICSETB)   Broker stub
ENTRY mainpgm
NAME ...
```

Linkage statements for Assembler applications:

```
INCLUDE cicslib(DFHEAI)  CICS Prolog Module
INCLUDE userlib(mainpgm) Main Program
INCLUDE cicslib(DFHEAI0) CICS Module
INCLUDE exxlib(CICSETB)  Broker stub
ENTRY mainpgm
NAME ...
```

COMETB

Scope

See *Available Stubs* above for overview of functionality and considerations.

Prerequisites and Installation Notes

- We recommend you load COMETB dynamically within the application program. COMETB is a non-reentrant member.
- COMETB is the stub for any Com-plete environment.
- COMETB must be available in the COMPLIB search chain if called dynamically. Modify the Com-plete procedure or job, adding the EXX load library to the COMPLIB library concatenations.
- COMETB requires about 950 KB storage above the line. Increase the Com-plete SYSPARM `THSIZEABOVE` by 950 KB.
- COMETB requires about 16 KB storage below the line. If the EXAENV data set is used and the `BLKSIZE` is greater than 6 KB, add the difference between 6 KB and the effective `BLKSIZE` to the recommended 16 KB. Increase the ULIB region size for your application that calls COMETB by this value.

Note:
For Natural, we recommend using stub *NATETB23*.

Linkage

It is also possible to link your application to member COMETB from the EntireX load library (EXX970.LOAD).

Simulating Environment Variables

Under Com-plete, a partitioned data set is used to store environment variables. The `SAGTOKEN` utility is not used here. See *EXAENV Environment Store*.

EXAENV Environment Store

Environment variables for the COMETB stub are handled by an easy-to-use approach that is compliant with Com-plete.

A partitioned data set is assigned by `DD EXAENV`. It represents the environment store for all Com-plete users. The member name is the name of the user logged on to Com-plete. If you want to define your own stub environment variables, add a text member with your user name and put all environment variables into it.

The environment store has following data set characteristics:

```
DSORG=PO
LRECL=80
RECFM=FB
```

A line in the text member setting the environment variable looks like:

```
ETB_STUBLOG=1
```

Variable name and variable value are left-justified and delimited by an equals sign. The first blank in the line identifies the end of the environment value definition.

The following keywords are currently supported:

- ETB_NONACT
- ETB_STUBLOG
- ETB_TIMEOUT

See *Environment Variables in EntireX*.

Note:
The destination of the stublog (environment variable `ETB_STUBLOG`) is controlled by the hardcopy setting in Com-plete. See hardcopy function from the `UUTIL` menu. If no device name has been defined, the stublog is displayed on the terminal. If `HC` has been specified, the stublog will be written to the Com-plete spool and routed to the device name supplied using `HC=name`.

IDMSETB

Scope

See *Available Stubs* above for overview of functionality and considerations.

Prerequisites and Installation Notes

None.

Linkage

Link your application to member IDMSETB from the EntireX load library (EXX970.LOAD).

Note:
IDMSETB cannot be called dynamically.

MPPETB

Scope

See *Available Stubs* above for overview of functionality and considerations.

Prerequisites and Installation Notes

- None.

Linkage

Link your application to member MPPETB from the EntireX load library (EXX970.LOAD) and the appropriate Adabas link module from the Adabas load library.

Note:
MPPETB can be called dynamically, but the appropriate ADALNK (ADALNI) must be linked to MPPTB beforehand.

NATETB23

Scope

See *Available Stubs* above for overview of functionality and considerations.

Prerequisites and Installation Notes

- NATETB23 is the recommended stub for Natural; you have just one stub for all environments (CICS, Batch, Com-plete).
- Set the Natural size parameters so that Natural can provide the stub with 34 KB at runtime.
- To use send/receive buffers of greater than 32 KB with NATETB23, WAL81 or above must be installed.

Prerequisites and Installation Notes for Natural

- Set the Natural sizes parameters so that Natural can provide the stub with 34 KB at runtime.
- Send/receive buffers of greater than 32 KB can be used with NATETB23 provided that WAL811 is installed.

Linkage

- To statically link the stub for use in z/OS Batch, TSO, CICS, IMS(BMP), IMS(MPP) and Com-plete, link NATETB23 into the load library where the shared Natural nucleus resides. NATETB23 is a reentrant and relocatable module.
- Alternatively, the NATETB23 stub can be dynamically invoked by the following Natural parameters:

  ```
  RCA=BROKER,RCALIAS=(BROKER,NATETB23)
  ```

Linkage under Natural

- Linking the stub for use in z/OS Batch, TSO, CICS, IMS(BMP), IMS(MPP) and Com-plete:

  Link NATETB23 in the Load Library to the shared part of the Natural nucleus. NATETB23 is a reentrant and relocatable module.
- To verify the installation, see *Installation Verification under Natural*.

Installation Verification under Natural

To verify the installation of the stub under Natural

1. Log on to Natural library SYSRPC and type MENU.
2. Invoke SM Service Directory Maintenance from the main menu.

3. Define the Node and Server and save.
4. Invoke XC Server Command Execution from the main menu for the node and server defined in the previous step.
5. Ping the server with the command PI.

Your environment and the stub are installed correctly if you receive

- `02150148 Connection error`, meaning that the broker and the RPC server are down;
- `00070007 Service not registered`, meaning that the broker is up and the RPC server is down;
- an answer from the RPC server.

For other return codes, see *Error Messages and Codes*.

Transport Methods for Broker Stubs

This section covers the following topics:

- Transport Method Values
- Default Transport Methods
- Setting the Timeout for the Transport Method
- Limiting the TCP/IP Connection Lifetime

Transport Method Values

The following table describes the possible values for the transport methods:

Transport Value	Description / Tips
NET	Use Adabas Cross-Memory Services as transport method. See *Installing Adabas Components for EntireX under z/OS* in the z/OS installation documentation. 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 V811 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 is *not* used, these same limits still apply under z/OS.
TCP	Use TCP/IP as transport method.
SSL	Use Secure Sockets Layer (SSL) as transport method.

Default Transport Methods

Stub	Default Transport Method
BROKER	NET
CICSETB	NET
COMETB	NET
IDMSETB	TCP
MPPETB	NET
NATETB23	NET

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 methods TCP and SSL, and is supported by all broker stubs under z/OS.

Notes:

1. Does not apply to stubs with transport method NET.
2. See *Setting a Transport Timeout with the SAGTOKEN Utility* below for more information.

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.

Setting a Transport Timeout with the SAGTOKEN Utility

To set a transport timeout with the SAGTOKEN utility

As a prerequisite, the steplib EXX970.LOAD of SAGTOKEN needs to be APF-authorized. For information on operand2, see the SAGTOKEN utility. For information on value, see the *Transport Timeout Values*.

- Use the SAGTOKEN utility with the following syntax:

```
//STEP EXEC PGM=SAGTOKEN,PARM=('SET LOCAL,TIMEOUT=value')
//STEPLIB DD DISP=SHR,DSN=EXX970.LOAD
```

Usage Example

```
//STEP EXEC PGM=SAGTOKEN,PARM=('SET LOCAL,TIMEOUT=20')
//STEPLIB DD DISP=SHR,DSN=EXX970.LOAD
```

Limiting the TCP/IP Connection Lifetime

With transport methods TCP/IP and SSL, the broker stub establishes one or more TCP/IP connections to the brokers specified with BROKER-ID. These connections can be controlled by the transport-specific CONNECTION-NONACT attribute on the broker side, but also by the transport-specific environment variable ETB_NONACT on the stub side. If ETB_NONACT is not 0, it defines the non-activity time (in

seconds) of active TCP/IP connections to any broker. See ETB_NONACT under *Environment Variables in EntireX*. Whenever the broker stub is called, it checks for the elapsed non-activity time and closes connections with a non-activity time greater than the value defined with ETB_NONACT.

Transport Non-activity Value	Description
0	Infinite lifetime until application is stopped.
n (seconds)	Transport connections with non-activity time greater than *n* will be closed.
Nothing set	Transport connections with non-activity time greater than 300s (default) will be closed.

Tracing for Broker Stubs

Scope

Setting tracing is supported by the broker stubs BROKER and CICSETB.

Load the appropriate stub or link your application with it from the EntireX load library (EXX970.LOAD). All stubs provide entry point BROKER.

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.

The trace level is set by assigning DDNAMEs EXALOG1, EXALOG2 or EXALOG3. The following DD statement instructs the stub to turn on tracing with level 3:

```
//EXALOG3 DD DUMMY
```

The DD statements represent a more efficient way of setting the trace level than the SAGTOKEN utility. However, the SAGTOKEN utility is still available. See *Setting Tracing with the SAGTOKEN Utility* below.

Trace Output Location

The location of the trace output file depends on the stub and the transport used:

Stub	Transport	DDNAME	Note
BROKER	SSL and TCP	DD:TRACE1	
	NET	DD:EXALOG1, DD:EXALOG2 or DD:EXALOG3	A real file assignment is required if the trace output is desired for transport NET.
CICSETB	all	DD:MSGUSR	

Examples

To set trace level 2 for stub BROKER with SSL or TCP transport:

```
//TRACE1 DD SYSOUT=*
//EXALOG2 DD DUMMY
```

To set trace level 3 for stub BROKER with NET transport:

```
//EXALOG3 DD SYSOUT=*
```

To set trace level 1 for stub BROKER with all transports:

```
//TRACE1  DD SYSOUT=*
//EXALOG1 DD SYSOUT=*
```

Setting Tracing with the SAGTOKEN Utility

Use the SAGTOKEN utility with the following syntax:

To switch on tracing

As a prerequisite, the steplib EXX970.LOAD of SAGTOKEN needs to be APF-authorized.

- The option to set trace is set with the SAGTOKEN utility using the following syntax:

  ```
  //STEP EXEC PGM=SAGTOKEN,PARM=('SET LOCAL,STUBLOG=value')
  //STEPLIB DD DISP=SHR,DSN=EXX970.LOAD
  ```

  Example:

  ```
  //STEP EXEC PGM=SAGTOKEN,PARM=('SET LOCAL,STUBLOG=1')
  //STEPLIB DD DISP=SHR,DSN=EXX970.LOAD
  ```

To switch off tracing

- Use the SAGTOKEN utility to set STUBLOG to zero.

  ```
  //STEP EXEC PGM=SAGTOKEN,PARM=('SET LOCAL,STUBLOG=0')
  //STEPLIB DD DISP=SHR,DSN=EXX970.LOAD
  ```

  Or:
  Use the SAGTOKEN utility to delete STUBLOG

```
//STEP EXEC PGM=SAGTOKEN,PARM=('DELETE LOCAL,STUBLOG')
//STEPLIB DD DISP=SHR,DSN=EXX970.LOAD
```

SVC Number for Broker Communication

Stub	Notes
BROKER	When Entire Net-Work transport is used, the default SVC number (249) can be overridden in the following ways: • By specifying the SVC number as part of the Broker ID, for example: `ETB220:SVC237:NET` • By including ADAUSER when linking the stub with the application. This enables the SVC number to be specified in the ADARUN cards of the application job. This option cannot be employed if the application operates within a multithreading application (multiple TCBs). • By using the supplied zap if neither of the above options is chosen. See *BROKER*.
CICSETB	When Entire Net-Work transport is used and the default SVC number (249) has to be changed, the Adabas communications module under CICS determines how the SVC number can be changed. See the Adabas documentation for information on how to change the SVC number.

COMETB	• The Adabas interface is integrated within the TP Monitor Com-plete.
	• Complete the following step:
	  ○ Add the ADASVC5 Parameter to Com-plete (or Adabas TPF) startup parameter as specified below:
	  `ADASVC5=(dbid,svc)`
	  ○ where <*dbid*> is the node ID selected for use by the EntireX Broker address space and <*svc*> is the Adabas SVC number used by the EntireX Broker started task/job.
	• Optionally consider how to specify the SVC number used for Broker communication maintained within Com-plete's Adabas interface. See the Com-plete documentation.

MPPETB	When Entire Net-Work transport is used and the default SVC number (249) has to be changed, the Adabas communications module under IMS/DC determines how the SVC number can be changed. See the Adabas documentation for information on how to change the SVC number.

Stub	Notes
NATETB23	● The following step must be completed if you are communicating with EntireX Broker from Com-plete or Adabas TPF: ○ Add the ADASVC5 Parameter to Com-plete (or Adabas TPF) startup parameter as specified below: `ADASVC5=(dbid,svc)` ○ where <*dbid*> is the node ID selected for use by the EntireX Broker address space and <*svc*> is the Adabas SVC number used by the EntireX Broker started task/job. ● For all other environments, see the Natural documentation for information on how to specify the SVC number used for Broker communication.

SAGTOKEN Utility

SAGTOKEN allows you to set variables. When setting variables with SAGTOKEN, SAGTOKEN error messages may be displayed on the operator console. The steplib EXX970.LOAD of SAGTOKEN must be APF-authorized. See *EntireX SAGTOKEN Messages*.

Syntax

```
//STEP EXEC PGM=SAGTOKEN,PARM=('operand1 operand2,operand3=value')
//STEPLIB DD DISP=SHR,DSN=EXX970.LOAD
```

Operands

operand1 is one of the following commands:

Command	Use
SET	Set or replace a SAGTOKEN variable.
DELETE	Delete a SAGTOKEN variable.
DISPLAY	Display a SAGTOKEN variable.

operand2 is either LOCAL or GLOBAL:

Value	Meaning
LOCAL	Applies to the address space.
GLOBAL	Applies to the z/OS image.

operand3 is the name of the variable to set:

Variable	Value
STUBLOG	For information on STUBLOG value, see *Tracing for Broker Stubs*.
TIMEOUT	For information on TIMEOUT value, see *Setting the Timeout for the Transport Method*.
TRANSPORT	For information on setting the transport mechanism, see ETB_TRANSPORT.

Note:
If a job uses SAGTOKEN to set local tokens in one step, we recommend that you delete these tokens prior to job termination in order to release all acquired resources.

Support of Clustering in a High Availability Scenario

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

- Introduction
- Exceptions
- Default
- Restriction

See also *High Availability in EntireX*.

Introduction

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

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

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

Exceptions

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

Default

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

Restriction

Support for this feature is currently not available under CICS, Com-plete, and IDMS/DC.

Considerations for Users without Adabas

For customers who do not have Adabas installed at their site,

- we recommend installing the Adabas modules delivered with EntireX in the library WAL826.
- the Adabas modules will greatly improve performance if the transport method NET is used and broker kernel and applications (client or server) communicating through the stub to the broker kernel on the same machine locally, see *Transport Methods for Broker Stubs*.

For information on how to install the Adabas SVC and install Adabas with TP Monitors, see *Installing Adabas Components for EntireX under z/OS* in the z/OS installation documentation.