webMethods EntireX

Installation under z/VSE

Version 10.5

October 2019

WEBMETHODS

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

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

Document ID: EXX-INSTALL-105-20220422VSE

Table of Contents

Installing EntireX under z/VSE

1 About this Documentation

Document Conventions

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

Online Information and Support

Product Documentation

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

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

Product Training

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

Tech Community

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

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

Product Support

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

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

Data Protection

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

2 General Information

Installation and system prerequisites for z/VSE and other EntireX platforms are described centrally. See *Prerequisites*. For communication using Entire Net-Work, Adabas SVC is required (delivered on medium in library WAL826.LIBR). See your Adabas documentation for details.

See also *Administering Broker Stubs under z/VSE*.

Contents of Installation Medium

The installation medium contains the following base files for z/VSE:

File	Library	Sublibrary	Description
EXP960.LIBR	SAGLIB	EXP960	RPC Server sublibrary.
EXX960.LIBR	SAGLIB	EXX960	EntireX sublibrary.
EXX960.LICS			License file.
EXX960.DC01			Readme file.
WAL826.LIBR	SAGLIB	WAL826	WAL sublibrary.
MLC127.LIBR	SAGLIB	MLC127	MLC sublibrary.

Migrating from EntireX Version 7.2.3

Perform the following steps to migrate from EntireX version 7.2.3:

- Change the Attribute File
- Check the License File
- Replace the EntireX Broker Stubs used by your Applications

Change the Attribute File

New functionality in EntireX version 9.6 led to changes and enhancements of various Broker attributes. This requires the attribute file to be brought up-to-date. A sample attribute file is delivered in `EXX960.LIBR(ETBnnn.ATR)`, which you can use as a basis. If, for example, dynamic memory management is turned on (default and recommended), all `NUM-*` parameters can be omitted. See *Broker-specific Attributes*.

Check the License File

Ensure the correct license file is referenced in your EntireX Broker startup job control. A sample startup job control is delivered in `EXX960.LIBR(RUNETB.J)`.

Replace the EntireX Broker Stubs used by your Applications

Although existing applications will still run after the migration of EntireX Broker, they should be relinked or replaced, if dynamically loaded, with stubs from version 9.6. See also *Administering Broker Stubs under z/VSE*.

3 Installing EntireX Broker under z/VSE

Prerequisites for EntireX Broker

Prerequisites for EntireX components are described centrally. See *z/VSE Prerequisites*.

For customers who do not have Adabas installed at their site, the Adabas modules delivered with EntireX in the library WAL826 for the Broker kernel must be installed.

For information on how to install Adabas SVC and configure Adabas cross memory services in batch and CICS, refer to your Adabas documentation.

Contents of Sublibrary EXX960

EntireX Broker, Configuration, Startup JCL and Executables

File	Description
RUNETB.J	EntireX Broker startup JCL.
PSFADA.J	Adabas COMPRESS and ADALOD sample JCL for persistent store creation.
ETB*nnn*.ATR	EntireX Broker Attribute File.
USRTCHA.A	Sample user translation routine.
ETBSTART.PHASE ETBNUC.PHASE ETBCEE.PHASE	EntireX Broker kernel.
EXXICU.PHASE	International components for Unicode.
USERSEC.PHASE SECUEXIO.PHASE ETBENC.OBJ ETBTB.OBJ ETBUEVA.OBJ ETBUPRE.OBJ ETBVEVA.OBJ ETBVPRE.OBJ	Security exit modules.
BKIMB.OBJ BKIMB.PHASE BROKERB.PHASE BKIMC.OBJ BKIMC.PHASE BROKERC.PHASE	EntireX Broker stubs.

Samples, Utilities and Sources

File	Description
RUNBCOC.J	Client sample JCL.
BCOC.PHASE	Client sample executable.
RUNBCOS.J	Server sample JCL.
BCOS.PHASE	Server sample executable.
RUNCMD.J	Command Services JCL.
ETBCMD.PHASE	Command Services executable.
RUNINFO.J	Information Services JCL.
ETBINFO.PHASE	Information Services executable.
BROKER.PRO CLIENT.PRO CLOGFLT.PRO CONV.PRO NET.PRO POOL.PRO PSF.PRO PSFADA.PRO RESOURCE.PRO SECURITY.PRO SERVER.PRO SERVICE.PRO STATIS.PRO TCP.PRO USER.PRO WKRUSAGE.PRO WORKER.PRO	Information Services Profiles.
COBDEF.C	COBOL ACI control block definitions.
COBINF.C	COBOL CIS definitions
ETBCDEF.H	C language ACI control block definitions.
ETBCINF.H	C language Command and Information Services API definitions.
ASMDEF.A	Assembler control block definitions.
ASMINF.A	Assembler CIS definitions.

EntireX Broker Stubs, Sample JCL, Objects and Executables

See *Administering Broker Stubs under z/VSE*.

File	Description
BKIMB.J	Sample JCL for linking the batch stub for use with external security
BKIMC.J	Sample JCL for linking the CICS stub for use with external security

Copying the Contents of the Installation Medium to Disk

Note: If you are using SMA, please refer to the System Maintenance Aid Manual, chapter Installing Software AG Products with SMA. If you are *not* using SMA, please follow the instructions below.

This section explains how to:

- Copy data set COPYTAPE.JOB from the installation medium to library.
- Modify this member to conform with your local naming conventions.

The JCL in this member is then used to copy all data sets from installation medium to disk.

If the data sets for more than one product are delivered on the installation medium, the member COPYTAPE.JOB contains the JCL to unload the data sets for all delivered products from the installation medium to your disk, except the data sets that you can directly install from the installation medium, for example, Natural INPL objects.

After that, you will have to perform the individual install procedure for each component.

Step 1: Copy Data Set COPYTAPE.JOB from Installation Medium to Disk

The data set COPYTAPE.JOB (file 5) contains the JCL to unload all other existing data sets from installation medium to disk. To unload COPYTAPE.JOB, use the following sample JCL:

```
* $$ JOB JNM=LIBRCAT,CLASS=0, +
* $$ DISP=D,LDEST=(*,UID),SYSID=1
* $$ LST CLASS=A,DISP=D
// JOB LIBRCAT
* ***************************************
* CATALOG COPYTAPE.JOB TO LIBRARY
* ***************************************
// ASSGN SYS004,NNN                        <------ tape address
// MTC REW,SYS004
// MTC FSF,SYS004,4
ASSGN SYSIPT,SYS004
```

```
// TLBL IJSYSIN,'COPYTAPE.JOB'
// EXEC LIBR,PARM='MSHP; ACC S=lib.sublib' <------- for catalog
/*
// MTC REW,SYS004
ASSGN SYSIPT,FEC
/*
/&
* $$ EOJ
```

where *NNN* is the installation medium address, and
lib.sublib is the library and sublibrary of the catalog

Step 2: Modify COPYTAPE.JOB

Modify `COPYTAPE.JOB` to conform with your local naming conventions and set the disk space parameters before submitting this job.

Step 3: Submit COPYTAPE.JOB

Submit `COPYTAPE.JOB` to unload all other data sets from the installation medium to your disk.

Preparing for Installation

Before the following steps can be performed, the EntireX installation medium must be copied to disk. If this has not yet been done, go to ***Step 1: Copy Data Set `COPYTAPE.JOB` from Installation Medium to Disk*** and follow the instructions there.

To use Software AG's System Maintenance Aid (SMA), you must modify SMA parameter values to suit your environment. Refer to the System Maintenance Aid Readme file (P060).

This installation procedure contains the following SMA jobs and steps:

Install Step	Activity	SMA Job	Job Step
1	*Adapt the SMA Parameters*	(P060)	
2	*Install the EntireX License File*	I070	7610
3	*Customize the Broker Attribute File*	I070	7604
4	*Customize the Broker Startup Job Control*	I070	7606
5	*Define an Adabas Persistent Store*	I050	7600 / 7610
6	*Install the Stubs*	-	-

Installing EntireX Broker

Step 1: Adapt the SMA Parameters

When using Software AG's System Maintenance Aid (SMA), you must modify SMA parameter values to suit your environment. Refer to the System Maintenance Aid readme file (P060).

Step 2: Install the EntireX License File

(SMA Job I070 / Step 7610)

The EntireX license file `EXX960.LICS` is delivered on the installation medium and/or CD.

The license file may reside in a sublibrary. The following FTP command sequence can be used to upload the file to z/VSE and place it into a sublibrary. Please note that the file needs to be uploaded in binary format and requires the record format "stream" on the target system.

```
ftp>
bin
quote site RECFM S
cd SAGLIB.EXX960
put <license_file> EXX960.LICS
```

The license file can then be referenced in the EntireX Broker startup procedure (`RUNETB.J`) by using a `SETPARM JCL` control statement.

```
// SETPARM ETBLIC=SAGLIB.EXX960(EXX960.LICS)
```

Alternatively, the JCL provided in file `EXX960.LIBJ` on the installation medium can be used to copy the license file to a sequential disk file. Make sure the sequential disk file is referenced in the EntireX Broker startup procedure (`RUNETB.J`).

```
// ASSGN  SYS002,DISK,VOL= VSEnnn,SHR
// DLBL   ETBLIC,'SAG.ETBLIC',0
// EXTENT SYS002 ,VSEnnn,,,xxxxx,yy
```

Step 3: Customize the EntireX Broker Attribute File (ETBnnn.ATR)

(SMA Job I070 / Step 7604)

Customize the delivered EntireX Broker attribute file (ETB*nnn*.ATR) to suit your environment. Specify at least the following four parameters:

```
BROKER-ID = ETBnnn
PORT   = port_number
ADASVC = svc_number
NODE   = node_number
```

Parameters `ADASVC` and `NODE` can be omitted with transport method TCP/IP (`TRANSPORT=TCP`). See *Broker Attributes*.

Please ensure the partition size is adequate for running EntireX Broker, taking into consideration the resource specified in the particular attribute file you are using. See also *Broker Resource Allocation*.

Step 4: Customize the EntireX Broker Startup Job Control (RUNETB.J)

(SMA Job I070 / Step 7606)

Customize the delivered EntireX Broker startup job control (`RUNETB.J`) to suit your environment. Ensure that license and attribute file are referenced correctly.

`RUNETB.J`

```
* $$ JOB JNM=RUNETB,CLASS=O,DISP=D
* $$ LST CLASS=A,DISP=H
// JOB RUNETB
*
* BROKER START UP JOB CONTROL
*
// LIBDEF *,SEARCH=(SAGLIB.EXX960,SAGLIB.MLC127,SAGLIB.WAL826)
/*
/* ATTRIBUTE FILE
/*
// SETPARM ETBATTR='DD:SAGLIB.EXX960(ETBnnn.ATR)'
/* / ASSGN SYS000,DISK,VOL=VSEnnn,SHR
/* / DLBL ETBATTR,'SAG.ETBATTR',0
/* / EXTENT SYS000,VSEnnn,,,xxxxx,yy
/*
/* TRACE DESTINATION
/*
// SETPARM ETBLOG='DD:SYSLST'
```

```
/* / ASSGN SYS001,DISK,VOL=VSEnnn,SHR
/* / DLBL ETBLOG,'SAG.ETBLOG',0
/* / EXTENT SYS001,VSEnnn,,,xxxxx,yy
/*
/* LICENSE FILE
/*
// SETPARM ETBLIC='DD:SAGLIB.EXX960(EXX960.LICS)'
/* / ASSGN SYS002,DISK,VOL= VSEnnn,SHR
/* / DLBL ETBLIC,'SAG.ETBLIC',0
/* / EXTENT SYS002 ,VSEnnn,,,xxxxx,yy
/*
/* TURN OFF CONSOLE PROMPT
/*
/* / UPSI 00000001
/*
// EXEC ETBSTART
// EXEC LISTLOG
/&
* $$ EOJ
```

If the subsequent optional installation steps are not required, you may now run job `RUNETB.J` to start the EntireX Broker.

To test your installation, see *Verifying the Installation of the Broker*.

Step 5: Define an Adabas Persistent Store (Optional)

(SMA Job I050 / Step 7600 / 7610)

To create an Adabas Persistent Store, adapt and run job `PSFADA.J` in sublibrary `EXX960`. Running this job will load an empty Adabas persistent store file into your Adabas database. To activate the Adabas persistent store, set up at least following parameters in the broker attribute file. See *Broker Attributes*.

```
DEFAULTS=BROKER
  STORE        = BROKER
  PSTORE       = HOT/COLD
  PSTORE-TYPE  = ADABAS
DEFAULTS=ADABAS
  DBID         = dbid
  FNR          = pstore_file_number
DEFAULTS=NET
  ADASVC       = svc_number
```

See *Managing the Broker Persistent Store* for more information.

Step 6: Install the Stubs

If you will be using the EntireX Broker stubs on z/VSE, see *Administering Broker Stubs under z/VSE*.

Verifying the Installation of the Broker

You can use the sample client and server programs `BCOC` and `BCOS` to test the EntireX Broker installation. First start the server program, then the client program.

Customize the delivered job control samples `RUNBCOS.J` and `RUNBCOC.J`. Choose the desired transport method and replace all placeholder values (in *`italic`* font).

Specifying the Broker ID

Depending on the transport method, the Broker ID can be specified in two 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

RUNBCOS.J

```
* $$ JOB JNM=RUNBCOS,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=H
// JOB RUNBCOS
*
* BCOS (SERVER) SAMPLE JCL
*
// LIBDEF *,SEARCH=(SAGLIB.EXX960,SAGLIB.WAL826)
/*
/* / EXEC BCOS,PARM='-b<ip>:<port>:TCP -i50 -p2000'
// EXEC BCOS,PARM='-bETB<nnn>:SVC<mmm>:NET -i50 -p2000'
/*
// EXEC LISTLOG
/&
* $$ EOJ
```

RUNBCOC.J

```
* $$ JOB JNM=RUNBCOC,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=H
// JOB RUNBCOC
*
* BCOC (CLIENT) SAMPLE JCL
*
// LIBDEF *,SEARCH=(SAGLIB.EXX960,SAGLIB.WAL826)
/*
/* / EXEC BCOC,PARM='-b<ip>:<port>:TCP -i50 -p2000'
// EXEC BCOC,PARM='-bETB<nnn>:SVC<mmm>:NET -i50 -p2000'
/*
// EXEC LISTLOG
/&
* $$ EOJ
```

Verification Steps

Run job `RUNBCOS.J`. It will register as `ASERVER`, `ASERVICE`, `ACLASS` at EntireX Broker. Run job `RUNBCOC.J` in a different z/VSE partition. This client sample program will exchange some test data with the previously registered server `ASERVER`.

RUNBCOS Sample Server Output

```
CLIENT: Message size 2000
CLIENT: Number of messages (10)
CLIENT: Broker ID localhost:1971:TCP
CLIENT: Using default class ACLASS
CLIENT: Using default server ASERVER
CLIENT: Using default service ASERVICE
CLIENT: Using default ACI version 10
CLIENT: Maximum stub ACI version: 10
CLIENT: Max kernel ACI version: 10
CLIENT: Kernel Version 9.6.0.00 Platform z/VSE 5.1.2
CLIENT: Using ACI version: 10
Request returns (900000000,838610000000000000000000000000)
Request returns (800000000,838610000000000000000000000000)
Request returns (700000000,838610000000000000000000000000)
Request returns (600000000,838610000000000000000000000000)
Request returns (500000000,838610000000000000000000000000)
Request returns (400000000,838610000000000000000000000000)
Request returns (300000000,838610000000000000000000000000)
Request returns (200000000,838610000000000000000000000000)
Request returns (100000000,838610000000000000000000000000)
Request returns (000000000,838610000000000000000000000000)
CLIENT: Test successfully performed
```

RUNBCOC Sample Client Output

```
SERVER: Userid EXXUSER
SERVER: Message size 2000
SERVER: Number of messages (10)
SERVER: Broker ID localhost:1971:NET
SERVER: Using default class ACLASS
SERVER: Using default server ASERVER
SERVER: Using default service ASERVICE
SERVER: Using default ACI version 10
SERVER: Maximum stub ACI version: 10
SERVER: Max kernel ACI version: 10
SERVER: Kernel Version 9.6.0.00 Platform z/VSE 5.1.2
SERVER: Using ACI version: 10
Incoming data (00000000000000000000000016838,000000009)
Incoming data (00000000000000000000000016838,000000008)
Incoming data (00000000000000000000000016838,000000007)
Incoming data (00000000000000000000000016838,000000006)
Incoming data (00000000000000000000000016838,000000005)
Incoming data (00000000000000000000000016838,000000004)
Incoming data (00000000000000000000000016838,000000003)
Incoming data (00000000000000000000000016838,000000002)
Incoming data (00000000000000000000000016838,000000001)
Incoming data (00000000000000000000000016838,000000000)
SERVER: Test successfully performed
```

Hints for Setting up Broker JCL in z/VSE

Where the Files Reside

- ETBLIC and ETBATTR can reside in z/VSE libraries as members (recommended) or they can reside as sequential disk files (SD files).
- ETBLOG, including possible traces, can go to SYSLST (recommended) or to an SD file.

Informing Broker where the Files Reside

- Supply the information with SETPARM JCL control statements

```
// SETPARM ETB(LIC|ATTR|LOG)='DD:dlbl_name'
```

for SD files, and for library members:

```
'DD:lib.sublib(member.type)'
```

- Supply the information as PARM in the EXEC card:

```
// EXEC ETBSTART,                      +
PARM='ENVAR("ETB_ATTR=DD:lib.sublib(member.type)")/'
```

If files reside in SD files, the necessary z/VSE JCL statements DLBL, EXTENT and ASSGN must be provided for each file too.

The shortest way

See also *Step 4: Customize the EntireX Broker Startup Job Control (RUNETB.J)*.

```
01 * $$ JOB JNM=RUNETB,CLASS=0,DISP=D
02 * $$ LST CLASS=A,DISP=H
03 // JOB RUNETB
04 // LIBDEF *,SEARCH=(SAGLIB.EXX960,SAGLIB.MLC127,SAGLIB.WAL826)
05 // UPSI 00000001
06 // SETPARM ETBLOG='DD:SYSLST'
07 // SETPARM ETBATTR='DD:SAGLIB.EXX960(ETBnnn.ATR)'
08 // SETPARM ETBLIC='DD:SAGLIB.EXX960(EXX960.LICS)'
09 // EXEC ETBSTART
10 // EXEC LISTLOG
```

```
11 /&
12 * $$ EOJ
```

To line 6:

This avoids message "4881I NO LABEL INFORMATION FOUND" during Broker startup.

To lines 7 and 8:

If z/VSE JCL parameters ETBATTR and ETBLIC are defined, the Broker retrieves the values of these parameters and derives the location of the attribute and license files from there.

To define ETBLIC, ETBATTR and ETBLOG as SD files

- ```
  // ASSGN  SYS000,DISK,VOL=VSEnnn,SHR
  // DLBL   ETBATTR,'SAG.ETBATTR',0
  // EXTENT SYS000 ,VSEnnn,,,xxxxx,yy
  /*
  // ASSGN SYS001,DISK,VOL= VSEnnn,SHR
  // DLBL ETBLOG,'SAG.ETBLOG',0
  // EXTENT SYS001,VSEnnn,,,xxxxx,yy
  /*
  // ASSGN  SYS002,DISK,VOL= VSEnnn,SHR
  // DLBL   ETBLIC,'SAG.ETBLIC',0
  // EXTENT SYS002 ,VSEnnn,,,xxxxx,yy
  /*
  ```

The following table describes the file assignment names:

File Assignment	DLBL Name	Description	See also Step
SYS000	ETBATTR	EntireX Broker attribute file.	*Step 3: Customize the EntireX Broker Attribute File (ETBnnn.ATR)*
SYS001	ETBLOG	Log file to which EntireX Broker writes trace output. Omitting this file assignment will route a trace output to SYSLST.	
SYS002	ETBLIC	EntireX License Certificate File.	
  ```

Using the BSI TCP/IP Stack

In addition to CSI's TCP/IP stack, EntireX Broker also supports the BSI TCP/IP stack. For this, the BSI library must be included in the `LIBDEF` search chain.

If multiple stacks are installed (for example, both CSI and the BSI stack are installed in parallel) specify the corresponding stack ID. See the sample job control for Broker startup below:

```
* $$ JOB JNM=RUNETB,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=H
// JOB RUNETB
*
* BROKER START UP JOB CONTROL
*
// LIBDEF *,SEARCH=(SAGLIB.EXX960,SAGLIB.MLC127,SAGLIB.WAL826,BSILIB.BSIvrs)
/*
/* ATTRIBUTE FILE
/*
// SETPARM ETBATTR='DD:SAGLIB.EXX960(ETBnnn.ATR)'
/* / ASSGN SYS000,DISK,VOL=VSEnnn,SHR
/* / DLBL ETBATTR,'SAG.ETBATTR',0
/* / EXTENT SYS000,VSEnnn,,,xxxxx,yy
/*
/* TRACE DESTINATION
/*
// SETPARM ETBLOG='DD:SYSLST'
/* / ASSGN SYS001,DISK,VOL=VSEnnn,SHR
/* / DLBL ETBLOG,'SAG.ETBLOG',0
/* / EXTENT SYS001,VSEnnn,,,xxxxx,yy
/*
/* LICENSE FILE
/*
// SETPARM ETBLIC='DD:SAGLIB.EXX960(EXX960.LICS)'
/* / ASSGN SYS002,DISK,VOL= VSEnnn,SHR
/* / DLBL ETBLIC,'SAG.ETBLIC',0
/* / EXTENT SYS002 ,VSEnnn,,,xxxxx,yy
/*
/* TURN OFF CONSOLE PROMPT
/*
/* / UPSI 00000001
/*
/*
/* SET STACK ID
/*
// OPTION SYSPARM='BSI_stack_id'
/*
// EXEC ETBSTART
// EXEC LISTLOG
/&
* $$ EOJ
```

4 Installing the z/VSE EntireX RPC Servers

This chapter covers the following topics:

For Natural RPC servers, see *Setting Up a Natural RPC Environment* in your Natural documentation.

Contents of Sublibrary EXP960

Configuration, Startup JCL and Server Executables

File	Description
RUNRPC.J	RPC Server for Batch startup JCL.
RPCPARM.CFG	RPC Server for Batch configuration file.
CICSDEF.J	CICS CSD definitions JCL.
VSAMDEF.J	Server mapping file VSAM definition JCL.
SVMPRIME.PHASE	Server mapping file VSAM prime utility.
RPCSRVB.PHASE RPCCONS.PHASE RPCLEST.PHASE	RPC Server for Batch.
EXXRFEC.PHASE EXXRFECA.PHASE EXXRFECS.PHASE EXXRFECX.PHASE	CICS Socket Listener.
RPCSRVC.PHASE	RPC Server for CICS.
COBUEXO2.C COBUEXO2.CPY	RPC Server for CICS user exit.

RPC Server for CICS ERXMAIN Control Block and RPC Online Maintenance Facility

File	Description
ERXMAIN.J	Assemble and LNKEDT JCL.
EMAINGEN.A	ERXMAIN control block macro.
ERXMAIN.A	Assembler source to generate the ERXMAIN control block containing the RPC Server for CICS default configuration.
ERXMAIN.PHASE	default ERXMAIN control block executable.
ERXMAINT.PHASE	RPC Online Maintenance Facility.
ERXMAPS.PHASE	RPC Online Maintenance Facility map.

Batch COBOL Client Server Example (CALC)

File	Description
CALCRUN.J	Batch CALC client startup JCL.
CALCCLT.PHASE	Batch CALC client executable.
CALC.PHASE	Batch CALC server program.

CICS COBOL Client Server Example (SQUARE)

File	Description
SQRECLT.PHASE	CICS SQUARE client executable.
SQREMAP.PHASE	CICS SQUARE client map.
SQUARE.PHASE	CICS SQUARE server program.

Installing the RPC Server for CICS

The EntireX RPC Server for z/VSE CICS® allows standard RPC clients to communicate with RPC servers on the operating system z/VSE under CICS. It supports the programming language COBOL. This section covers the following topics:

- Step 1: Define a Server-side Mapping Container - VSAMDEF.J (Optional)
- Step 2: Build the ERXMAIN Control Block (Optional)
- Step 3: Modify CICS Startup JCL
- Step 4: Update the CICS Tables
- Step 5: Start the RPC Server for CICS
- Installing Multiple EntireX RPC Servers in the same CICS (Optional)

Note: Installing the RPC Server for CICS also installs the CICS Socket Listener used by the EntireX Adapter or RPC Server for CICS Socket Listener. For more information and configuration details see *Preparing for CICS Socket Listener* (EntireX Adapter | RPC Server for CICS Socket Listener).

Step 1: Define a Server-side Mapping Container - VSAMDEF.J (Optional)

If you are using or plan to use server-side mapping files, you need to set up a server-side mapping container. A server-side mapping file is a Designer file with extension .svm. See *Server Mapping Files for COBOL*. If this step is omitted, the RPC server will start without the server-side mapping container. This means that server programs cannot make use of special COBOL syntax and features. See *When is a Server Mapping File Required?* in the Designer documentation.

(SMA Job I008 / Step 7570 and Job I025 / Step 7570)

The server-side mapping container stores the content of server-side mapping files, which are used at runtime to marshal and unmarshal the RPC data stream. This enables the RPC server to support special COBOL syntax. The server-side mapping container is technically a VSAM file that needs to be defined and initialized.

Note: Perform this step for RPC Server for Batch and RPC Server for CICS with different file names.

Customize and run job `VSAMDEF.J`. Specify the desired VSAM KSDS cluster name, catalog and catalog ID. These are marked by the placeholders "`<...>`" in `VSAMDEF.J`.

See also *Server-side Mapping Files*.

SVM File Definition Job Control

```
* $$ JOB JNM=VSAMDEF,CLASS=H,DISP=D
* $$ LST CLASS=0,DISP=D
// JOB VSAMDEF
*
* DEFINE AND INITIALIZE THE RPC SVM FILE VSAM CLUSTER
*
// EXEC IDCAMS,SIZE=AUTO
 /* ------------------------------------------------------------ */
 /*  DELETE SERVER MAPPING VSAM CLUSTER                          */
 /* ------------------------------------------------------------ */
   DELETE (<vsam_ksds_cluster_name>) CL NOERASE PURGE   -
   CATALOG(<catalog_name>)
   SET MAXCC = 0
   SET LASTCC = 0
 /* ------------------------------------------------------------ */
 /*  DEFINE SERVER MAPPING VSAM CLUSTER                          */
 /* ------------------------------------------------------------ */
   DEFINE CLUSTER                                        -
          ( NAME(<vsam_ksds_cluster_name>)               -
            RECORDSIZE(1024 16384)                       -
            RECORDS(1000 1000)                           -
            KEYS(255 0)                                  -
            INDEXED                                      -
            SHR(2,3)                                     -
```

```
            VOL(<volume>)                                  -
          )                                                -
          DATA(NAME(<vsam_ksds_cluster_name>.DATA))        -
          INDEX(NAME(<vsam_ksds_cluster_name>.INDEX))      -
          CATALOG(<catalog_name>)
/*
/* ------------------------------------------------------------ */
/*  INITIALIZE SERVER MAPPING VSAM CLUSTER                      */
/* ------------------------------------------------------------ */
// DLBL ERXSVM,'<vsam_ksds_cluster_name>',0,VSAM,CAT=<catalog>
// LIBDEF *,SEARCH=(SAGLIB.EXP960),TEMP
// EXEC SVMPRIME,SIZE=AUTO
/*
/&
* $$ EOJ
```

Step 2: Build the ERXMAIN Control Block (Optional)

(SMA Job I070 / Step 7572, 7573)

The ERXMAIN control block holds the EntireX RPC Server for CICS parameter settings. This configuration can be manually maintained by using the *RPC Online Maintenance Facility*. A preconfigured `ERXMAIN` phase is delivered in sublibrary EXP960. To alter this configuration, perform the following steps:

1. Adapt source `EMAINGEN.A` to match your environment. See *Customizing the RPC Server*.
2. Adapt job `ERXMAIN.J` to match your environment.
3. Run job `ERXMAIN.J`.

Running job `ERXMAIN.J` assembles macro `ERXMAIN` and links a new `ERXMAIN` phase with the preferred default settings.

The name of the phase may be altered within job `ERXMAIN.J`. This allows you to run multiple EntireX RPC Server for CICS instances. See ***Installing Multiple EntireX RPC Servers in the same CICS (Optional)***.

Step 3: Modify CICS Startup JCL

To enable CICS to find the various programs defined to the PPT (installation step 4), the CICS startup JCL needs to be modified to include the sublibraries EXP960 and EXX960 into the DFHRPL chain.

Additionally, if installation Step 1 was performed, a DLBL statement, as shown below, needs to be added for the VSAM KSDS cluster of the SVM file.

```
// DLBL ERXSVM,'<vsam_ksds_cluster_name>',0,VSAM,CAT=<catalog>
```

Each RPC Server instance running in same CICS partition requires its own SVM file, that is, its own DLBL entry. You can choose any DLBL name. ERXSVM is the default. It can be altered manually in the RPC Online Maintenance Facility (see *RPC Online Maintenance Facility* in the RPC Server for CICS documentation) or set up permanently in the ERXMAIN control block (installation Step 2).

Step 4: Update the CICS Tables

(SMA Job I005 / Step 7570)

Add the definitions listed below to your CICS system, either manually or with job `CICSDEF.J` in sublibrary EXP960.

- File Definitions
- Program Definitions
- Mapset Definition
- Transaction Definitions

File Definitions

If installation Step 1 was performed, the SVM file VSAM KSDS cluster needs to be defined. Replace the *<vsam_ksds_cluster_name>* in the `DEFINE` statement.

```
DEFINE FILE(ERXSVM) GROUP(EXX)
  DSNAME(<vsam_ksds_cluster_name>)
  DESCRIPTION(ENTIREX RPC SVM FILE)
  LSRPOOLID(NONE) ADD(YES) BROWSE(YES) DELETE(YES) READ(YES)
  UPDATE(YES) RECORDSIZE(16384) KEYLENGTH(255) STRINGS(2)
  DATABUFFERS(201) INDEXBUFFERS(200)
```

Program Definitions

```
DEFINE PROGRAM(BKIMC) GROUP(ERX) LANGUAGE(ASSEMBLER)
DEFINE PROGRAM(BROKERC) GROUP(ERX) LANGUAGE(C)
DEFINE PROGRAM(ERXMAIN) GROUP(ERX) LANGUAGE(ASSEMBLER)
DEFINE PROGRAM(ERXMAINT) GROUP(ERX) LANGUAGE(C)
DEFINE PROGRAM(RPCSRVC) GROUP(ERX) LANGUAGE(C)
DEFINE PROGRAM(SQRECLT) GROUP(ERX) LANGUAGE(COBOL)
DEFINE PROGRAM(SQUARE) GROUP(ERX) LANGUAGE(COBOL)
```

The programs in the definitions above are used for following purposes:

Program	Purpose
BKIMC	CICS Broker stub interface.
BROKERC	CICS Broker stub.
ERXMAIN	Control block.
ERXMAINT	RPC Online Maintenance Facility.
RPCSRVC	RPC Server for CICS.
SQRECLT	COBOL CICS SQUARE example client.
SQUARE	COBOL CICS SQUARE example server.

Mapset Definition

```
DEFINE MAPSET(ERXMAPS) GROUP(EXP)
DEFINE MAPSET(SQREMAP) GROUP(EXP)
```

This mapset is used by the RPC Online Maintenance Facility. See *RPC Online Maintenance Facility* in the RPC Server for CICS documentation.

Transaction Definitions

```
DEFINE TRANSACTION(ERXM) GROUP(EXXERX) PROGRAM(ERXMAINT) TWASIZE(28)
DEFINE TRANSACTION(ESRV) GROUP(EXXERX) PROGRAM(RPCSRVC)  TWASIZE(28)
DEFINE TRANSACTION(EC01) GROUP(ERX) PROGRAM(SQRECLT)  TWASIZE(28)
```

Transaction ERXM is used to run the *RPC Online Maintenance Facility*.
Transaction ESRV is used to run the RPC Server for CICS; see *Customizing the RPC Server*.

Note: If required, adapt CICS settings, for example TWASIZE. See *CICS Settings*.

Step 5: Start the RPC Server for CICS

To start the RPC Server for CICS, use the RPC Online Maintenance Facility by entering transaction ERXM (see *RPC Online Maintenance Facility* in the RPC Server for CICS documentation). The RPC Online Maintenance Facility is command-line oriented. If the default ERXMAIN control block is used, adapt the parameter settings to match the environment - especially the Broker ID - by entering:

```
BROKER=<Broker ID>
```

Depending on the communication method, the Broker ID can be specified in two 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

To start RPC Server for CICS, press PF8.

To stop RPC Server for CICS, press PF10.

Installing Multiple EntireX RPC Servers in the same CICS (Optional)

To install a second RPC server in the same CICS

1 Copy the default RPC Server for CICS transaction definition ESRV and give it a unique name, e.g. ESR2.

```
CEDA COPY TRANSACTION(ESRV) GROUP(ERX) TO(ERX2) AS(ESR2)
```

2 Copy the default RPC Server for CICS *ERXMAIN Control Block* and give it a unique name, e.g. ERXMAIN2.

```
CEDA COPY PROGRAM(ERXMAIN) GROUP(ERX) TO(ERX2) AS(ERXMAIN2)
```

3 Add the new group ERX2 to the CICS autoinstall list.

```
CEDA ADD GROUP(ERX2) LIST(listname) AFTER(groupname)
```

4 Build a new *ERXMAIN Control Block* and give it the name created above, e.g. ERXMAIN2.

As a minimum, set the *ERXMAIN Macro* parameter REPL in the ERXMAIN Control Block to the new RPC server transaction ID created above, e.g. REPL=ESR2.

The second RPC Server for CICS can now be started. See *Starting the RPC Server* under *RPC Online Maintenance Facility* in the RPC Server for CICS documentation.

Installing the RPC Server for Batch

The EntireX RPC Server for z/VSE Batch allows standard RPC clients to communicate with RPC servers on the operating system z/VSE under Batch. It supports the programming language COBOL and works together with the *COBOL Wrapper* and *IDL Extractor for COBOL*. This section covers the following topics:

Step 1: Define a Server-side Mapping Container - VSAMDEF.J (Optional)

If you are using or plan to use server-side mapping files, you need to set up a server-side mapping container. A server-side mapping file is a Designer file with extension .svm. See *Server Mapping Files for COBOL*. If this step is omitted, the RPC server will start without the server-side mapping container. This means that server programs cannot make use of special COBOL syntax and features. See *When is a Server Mapping File Required?* in the Designer documentation.

(SMA Job I008 / Step 7570)

The server-side mapping container stores the content of server-side mapping files, which are used at runtime to marshal and unmarshal the RPC data stream. This enables the RPC server to support special COBOL syntax. The server-side mapping container is technically a VSAM file that needs to be defined and initialized.

Note: Perform this step for RPC Server for Batch and RPC Server for CICS with different file names.

Customize and run job `VSAMDEF.J`. Specify the desired VSAM KSDS cluster name, catalog, catalog ID and volume. These are marked by the placeholders "`<...>`" in `VSAMDEF.J`.

See also *Server-side Mapping Files*.

SVM File Definition Job Control

```
* $$ JOB JNM=VSAMDEF,CLASS=H,DISP=D
* $$ LST CLASS=0,DISP=D
// JOB VSAMDEF
*
* DEFINE AND INITIALIZE THE RPC server mapping file VSAM CLUSTER
*
// EXEC IDCAMS,SIZE=AUTO
 /* ------------------------------------------------------------ */
 /*  DELETE SERVER MAPPING VSAM CLUSTER                          */
```

```
 /* ------------------------------------------------------------ */
   DELETE (<vsam_ksds_cluster_name>) CL NOERASE PURGE   -
   CATALOG(<catalog_name>)
   SET MAXCC = 0
   SET LASTCC = 0
 /* ------------------------------------------------------------ */
 /*  DEFINE SERVER MAPPING VSAM CLUSTER                          */
 /* ------------------------------------------------------------ */
   DEFINE CLUSTER                                        -
          ( NAME(<vsam_ksds_cluster_name>)               -
            RECORDSIZE(1024 16384)                       -
            RECORDS(1000 1000)                           -
            KEYS(255 0)                                  -
            INDEXED                                      -
            SHR(2,3)                                     -
            VOL(<volume>)                                -
          )                                              -
          DATA(NAME(<vsam_ksds_cluster_name>.DATA))      -
          INDEX(NAME(<vsam_ksds_cluster_name>.INDEX))    -
          CATALOG(<catalog_name>)
/*
/* ------------------------------------------------------------ */
/*  INITIALIZE SERVER MAPPING VSAM CLUSTER                      */
/* ------------------------------------------------------------ */
// DLBL ERXSVM,'<vsam_ksds_cluster_name>',0,VSAM,CAT=<catalog>
// LIBDEF *,SEARCH=(SAGLIB.EXP960),TEMP
// EXEC SVMPRIME,SIZE=AUTO
/*
/&
* $$ EOJ
```

Step 2: Adapt the RPC Parameter File

(SMA Job I200 / Step 7570)

`RPCPARM.CFG` in sublibrary EXP960 contains the RPC server parameters. If the default settings are used, only the `BROKERID` parameter needs to be set up according to your environment. The RPC Server for Batch will then run in a default configuration with two permanent worker tasks (up to 16 replicates are possible) and the deployment service turned off. See *Server Mapping Deployment Wizard*. It will register at EntireX Broker as service SRV1, class RPC and service CALLNAT, which corresponds to the settings in the delivered Broker attribute file. Security is turned off. See also *Configuring the RPC Server*.

Depending on the communication method, the Broker ID can be specified in two 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

To turn on the Deployment Service, a built-in service of the RPC server, (see *Deployment Service*) uncomment the line shown below. This will allow deploying information from the *Designer* into the SVM file. Additionally, the SVM file needs to be installed; See ***Step 1: Define a Server-side Mapping Container - VSAMDEF.J (Optional)***.

```
* DEPLOYMENT=YES                                register Deployment Service
```

The default configuration file RPCPARM.CFG:

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                           *
*               EntireX RPC Server configuration file                       *
*                                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
* * * * * * * * * * EntireX Broker Parameters * * * * * * * * * * * * * *
*
BROKERID=<ip>:<port>:TCP                    Broker ID if TCP/IP is used
* BROKERID=ETB<nnn>:SVC<ccc>:NET          Broker ID if XCOM is used
*
SERVERNAME=SRV1
SERVICE=CALLNAT
CLASS=RPC
*
TIMEOUT=30
*
* USERID=ERXUSER
* PASSWORD=PASSWORD
* LOGON=YES
*
* * * * * * * * * * EntireX RPC Server Parameter  * * * * * * * * * * * *
*
```

```
RESTARTCYCLES=3                          try three times (1 minute
*                                        interval) when EntireX
*                                        Broker is unavailable
*
* DEPLOYMENT=YES                          register Deployment Service
*
* TRACELEVEL=ADVANCED                     NONE, STANDARD or ADVANCED
*
* ETBLNK=BROKER                           required if Relay Manager
*                                        communication is desired
*
* Start up a fixed number of workers      (up to 16 workers possible)
* ------------------------------------------------------------------
MINWORKERS=2                             number of parallel server
*                                        replicates started
*                                        MAXWORKERS will be ignored
ENDWORKERS=NEVER                         stop never
*
*
* Balance the load of available workers
* ------------------------------------------------------------------
* MINWORKERS=0                            keep alive at least one
*                                        replicate
* MAXWORKERS=<n>                          n .. maximum of parallel
*                                        server replicates
* ENDWORKERS=IMMEDIATLY                   stop immediately after
*                                        request worked off
*
* Balance the load of available workers
* ------------------------------------------------------------------
* MINWORKERS=1                            keep alive at least one
*                                        replicate
* MAXWORKERS=<n>                          n..maximum of parallel
*                                        server replicates
* ENDWORKERS=TIMEOUT                      stop after timeout received
*
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Step 3: Customize the Startup JCL - RUNRPC.J

Adapt the RPC Server for Batch startup JCL `RUNRPC.J` to fit your environment.

If installation Step 1 was performed, include the VSAM KSDS Cluster name of the SVM file and uncomment following line:

```
/* / DLBL ERXSVM,'<vsam_ksds_cluster_name>',0,VSAM,CAT=<catalog>
```

Insert the sublibraries containing your own server programs into the `LIBDEF` search chain.

If you wish to suppress the console prompt, uncomment the UPSI statement.

```
/* / UPSI 00000001
```

Note: In the current RPC Server version, the UPSI setting suppresses the startup of the RPC Server Console task. In that case an `MSG partition_id` command will not have any effect as there is no AR routine present. Stopping the RPC Server is then only possible with `ETBCMD`.

RPC Server for Batch Job Control:

```
* $$ JOB JNM=RUNRPC,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=H
// JOB RUNRPC
*
* BATCH RPC SERVER START UP JOB CONTROL
*
// OPTION DUMP,NOSYSDUMP
/*
/* TURN OFF RPC SERVER CONSOLE PROMPT
/*
/* / UPSI 00000001
/*
/* SERVER MAPPING FILE
/*
/* / DLBL ERXSVM,'<vsam_ksds_cluster_name>',0,VSAM,CAT=<catalog>
/*
// LIBDEF PHASE,SEARCH=(SAGLIB.EXP960,                                  +
               SAGLIB.EXX960,                                         +
               SAGLIB.WAL826),TEMP
/*
// EXEC RPCSRVB,SIZE=AUTO,PARM='CFG=DD:SAGLIB.EXP960(RPCPARM.CFG)'
/*
/&
* $$ EOJ
```

Step 4: Start RPC Server

To start the RPC Server for Batch

- Run job `RUNRPC.J`.

To stop the RPC Server for Batch

- Use the following console command:

```
task_id STOP
```

Verifying the Installation

There are two COBOL client/server examples delivered as executables with the RPC servers, one for use in Batch (`CALC`), the other for CICS (`SQUARE`). These examples are useful for verifying the RPC Server installation. They are a subset of the z/VSE COBOL examples provided with EntireX on Windows. See *Client and Server Examples for z/VSE CICS* and *Client and Server Examples for z/VSE Batch* for more information.

This section covers the following topics:

- CICS COBOL SQUARE Client Server Example (Optional)
- Batch COBOL CALC Client Server Example (Optional)

CICS COBOL SQUARE Client Server Example (Optional)

This COBOL client server example is delivered as executable and ready to run. Its purpose is to verify the RPC Server for CICS installation.

Ensure following CICS transaction and programs are defined:

```
DEFINE TRANSACTION(EC01) GROUP(ERX) PROGRAM(SQRECLT) TWASIZE(28)
DEFINE PROGRAM(SQRECLT) GROUP(ERX) LANGUAGE(COBOL)
DEFINE PROGRAM(SQUARE) GROUP(ERX) LANGUAGE(COBOL)
```

Executing the transaction `EC01` will display following CICS map:

```
16:06:11       ---- RPC  SQRECLT   example for  COBOL  ----       31/01/14

BROKER-ID localhost:1971

CLASS     RPC                              USER-ID   COBUSER
SERVER    SRV1                             PASSWORD
SERVICE   CALLNAT

Operand   000000012




--------------------------------------------------------------------------------
 F3 Exit  F5 Exec
```

Enter the Broker ID, depending on the transport method used:

- **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

Enter the BROKER-ID and press **PF5** to execute the client.

Batch COBOL CALC Client Server Example (Optional)

This COBOL client server example is delivered as executable and ready to run. Its purpose is to verify the RPC Server for Batch installation.

Customize CALCRUN.J to fit the environment. Set up BROKERID as described under *CICS COBOL SQUARE Client Server Example (Optional)*.

Run job CALCRUN.J.

```
* $$ JOB JNM=CALCRUN,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=H
// JOB CALCRUN
*
* CALC EXAMPLE CLIENT START UP JOB CONTROL
*
// LIBDEF PHASE,SEARCH=(SAGLIB.EXP960,                              +
               SAGLIB.EXX960,                                      +
               SAGLIB.WAL826),TEMP
// EXEC  CALCCLT,SIZE=AUTO
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Input Parameter                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
BROKERID <ipaddr>:<port>:TCP                                      *
* BROKERID ETB<nnnnn>::NET                                        *
* USERID   <userid>                                               *
* PASSWORD <password>                                             *
```

```
CLASS    RPC                                                          *
SERVER   SRV1                                                         *
SERVICE  CALLNAT                                                      *
LOGON                                                                 *
CALC     + 00012345 00067890                                          *
CALC     - 00067890 00012345                                          *
CALC     * 00001234 00005678                                          *
CALC     / 00005678 00001234                                          *
CALC     % 00005678 00001234                                          *
END                                                                   *
/*
/&
* $$ EOJ
```

5 Installing EntireX Security under z/VSE

Installing EntireX Security for Broker Kernel

This section describes the steps for installing EntireX Security for Broker kernel under z/VSE.

Step 1: Modify Broker Attribute File

1. Insert the following parameter in the section `DEFAULTS=BROKER` of the Broker attribute file:

```
SECURITY=YES
```

2. Modify the `SECURITY-PARMS` parameter according to your requirements: see *EntireX Security*.

Note: Setting `SECURITY=YES` will load the provided `PHASE` module `USRSEC` from the `EXX960` product sublibrary. This module will issue the IBM `PRODID` macro in order to obtain authorization to perform privileged operations, such as execute the `RACROUTE`.

Step 2: Configure IBM Basic Security Manager

The IBM Basic Security Manager must be installed and configured on your system in order to use EntireX Security. This allows EntireX Broker to perform authentication based on a user ID and password stored in BSM for all connected application components. See IBM documentation *z/VSE Installation*, *z/VSE Administration* and *z/VSE Planning* for complete details of IBM Basic Security Manager (BSM).

Note: It is not necessary to activate the Basic Security Manager in batch.

Step 3: Define User Profiles for Basic Security Manager

See your IBM documentation for detailed instructions for defining user IDs for Basic Security Manager (BSM). The section *Resource Definition* with subsection *Tailoring the Interactive Environment/Maintaining User Profiles* provides a description of both online (ICCF) and batch utilities for adding user IDs. The online ICCF transaction for adding user IDs is shown below.

Required fields are shown in **bold**. If `DAYS=0`, the password will never expire.

Alternatively, you can use the batch program `IESUPDCF`. See your z/VSE documentation for details.

```
    ↩
_____________________________________________________________________________
   |                                                                          ↩
    |
   |  IESADMUPBA                ADD OR CHANGE USER PROFILE                    ↩
    |
   |  Base     II       CICS     ResClass ICCF                                 ↩
    |
   |                                                                          ↩
    |
   |  To CHANGE, alter any of the entries except the userid.                  ↩
    |
   |                                                                          ↩
    |
   |    USERID..... ENDU       4 - 8 characters (4 characters for ICCF ↩
users) |
   |                                                                          ↩
    |
   |    INITIAL PASSWORD... ________   3 - 8 characters                       ↩
    |
   |                                                                          ↩
    |
   |    DAYS..... 000       0-365 Number of days before password expires ↩
  |
   |    REVOKE DATE........ ________   Date when Userid will be revoked (mm/dd/yy) ↩
    |
   |                                                                          ↩
    |
   |    USER TYPE.......... 1          1=Administrator, 2=Programmer, 3=General   ↩
    |
   |    INITIAL NAME....... IESEADM    Initial function performed at signon       ↩
    |
   |    NAME TYPE.......... 2          1=Application, 2=Selection Panel           ↩
    |
   |    SYNONYM MODEL...... ________   Userid to be used as model for synonyms    ↩
    |
   |                                                                          ↩
    |
   |                                                                          ↩
    |
   |                                                                          ↩
    |
   |                                                                          ↩
    |
   |  PF1=HELP                    3=END                    5=UPDATE            ↩
    |
   |                 8=FORWARD                                                ↩
    |
   |                                                                          ↩
    |
   |                                                                          ↩
    |
```

```
    ↩
|_____________________________________________________________________|
```

Step 4: Start / Restart Broker Kernel

This is needed to pick up changes to the Broker attribute file and to initialize Broker kernel under z/VSE as an authorized subsystem able to perform security checks.

Installation of EntireX Security for Broker kernel is now complete.

Setting up EntireX Security for Broker Stubs

This section describes the steps for installing EntireX Security for Broker stubs under z/VSE.

The delivered phases `BKIMB.PHASE` and `BKIMC.PHASE` are linked for use with internal security, which requires an application to use ACI version 8 or above. If you are running your application at ACI version 7 or below, the steps above are required to install EntireX Security for the Broker stubs in all environments where applications execute. These steps are *not* required if you are running your application at ACI version 8 or above.

Step 1: Relink the Stub Modules or your Application for Use with External Security

To enable external security, relink the stub modules `BKIMB.PHASE`, and `BKIMC.PHASE` (and your application if it does not dynamically load the stub).

Additionally include the following objects:

- `ETBUEVA`
- `ETBUPRE`
- `ETBVPRE`
- `ETBVEVA`
- `ETBENC`
- `ETBTP`

The following job control is delivered and may be used for relinking the various stub modules:

- `BKIMB.J`
- `BKIMC.J`

Additionally, a detailed description of how to link the stubs and your application can be found under *Administering Broker Stubs under z/VSE*.

Step 2: Rename SECUEXI0

Rename the phase `SECUEXIO.PHASE` in library EXX960 to `SECUEXIT.PHASE`.

6 EntireX CICS Socket Listener

Prerequisites for all EntireX components are described centrally. See *z/VSE Prerequisites* .

Installation under z/VSE

Execute job `CICSDEF.J` in sublibrary EXP960. This will create all relevant CSD entries for the CICS Socket Listener. This job:

- installs the load modules `EXXRFEC`, `EXXRFECX`, `EXXRFECA` and `EXXRFECS`
- creates the following CICS CSD definitions:
  - program definitions for the load modules `EXXRFEC`, `EXXRFECX`, `EXXRFECA` and `EXXRFECS`, where definition for `EXXRFECS` needs `EXECKEY(CICS)`
  - transaction XRFE, which is the standard transaction name that can be maintained with the RPC Server or Adapter configuration. If you need to change this CSD entry default, you also need to adapt the configuration parameter `CICS transaction ID`. More information:
    - if you are using the EntireX Adapter, see *Connection Parameters for CICS Socket Listener Connections*
    - if you are using the RPC Server for CICS Socket Listener, see *Configuring an RPC Server Instance* > *CICS* using the Command Central GUI | Command Line

To run the CICS programs in a separate user transaction

1 Define your user transaction ID in the RPC Server for CICS Socket Listener, for example:

```
cics.sl.user.transaction.id=UTSK
```

For more information see *Configuring the CICS Socket Listener Side* in the RPC Server for CICS Socket Listener documentation.

2 Create the CSD for the user transaction ID.

3 In this CSD definition, specify `EXXRFECU` for the `PROGRAM` attribute. To define, for example, `UTSK` as user transaction ID, use the following commands:

```
DEFINE TRANSACTION(UTSK) GROUP(...)
DESCRIPTION(CICS Socket Listener user transaction)
PROGRAM(EXXRFECU)
```

Note: The CICS Socket Listener is also installed along with the RPC Server for CICS. See ***Installing the RPC Server for CICS***.

Configuration

For configuration, refer to *Preparing for CICS Socket Listener* (EntireX Adapter | RPC Server for CICS Socket Listener).