# webMethods EntireX

## EntireX RPC Server for CICS® ECI

Version 10.8

October 2022

WEBMETHODS

This document applies to webMethods EntireX Version 10.8 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-CICSECI-108-20220601**

# Table of Contents

# 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:<br>Variables for which you must supply values specific to your own situation or environment.<br>New terms the first time they occur in the text.<br>References to other documentation sources. |
| `Monospace font` | Identifies:<br>Text you must type in.<br>Messages displayed by the system.<br>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.

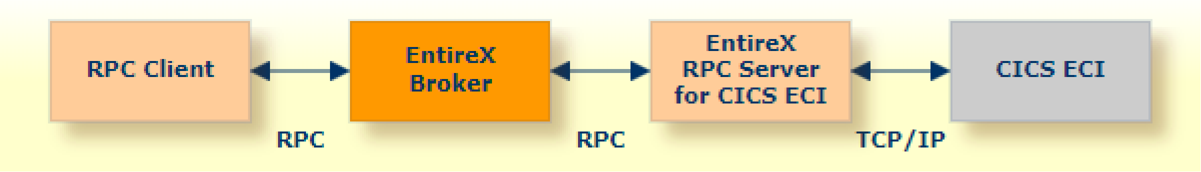
## Data Protection

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 Introduction to the RPC Server for CICS ECI

## Overview

The RPC Server for CICS ECI acts on one side as an RPC server and on the other side as a client for CICS ECI. The RPC Server for CICS ECI is a Java-based component that can run on a different host to the one where CICS is running. This allows it to operate with a zero footprint of EntireX on the CICS host.



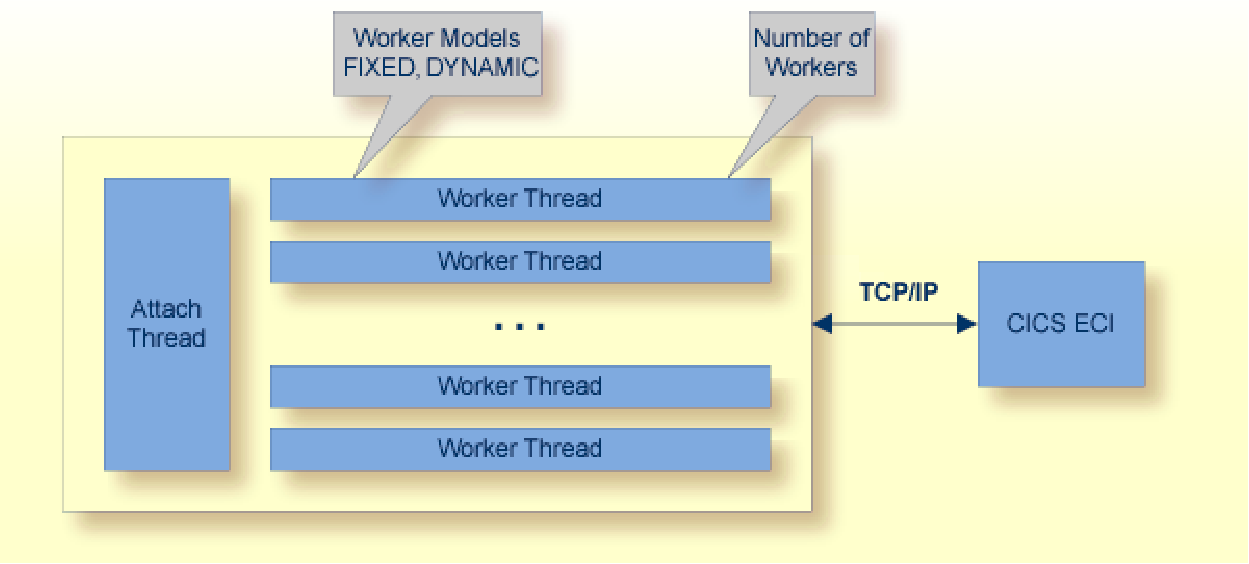
For local extraction, all source files have to be stored locally on the same machine where the Designer is running.

- For existing CICS COBOL programs, use the *IDL Extractor for COBOL* to extract the *Software AG IDL File* in the IDL Editor documentation for the RPC clients.
- For existing CICS PL/I programs, use the *IDL Extractor for PL/I* to extract the *Software AG IDL File* in the IDL Editor documentation for the RPC clients.

Remote extraction requires an RPC server running under z/OS with Extractor Service (Batch | IMS).

- For COBOL, see *Step 2: Select a COBOL Extractor Environment or Create a New One* in the IDL Extractor for COBOL documentation.
- For PL/I, see *Extract Software AG IDL File from a Remote PL/I RPC Environment* in the IDL Extractor for PL/I documentation.

## Worker Models



RPC requests are worked off inside the RPC server in worker threads. If you are using RPC conversations, each RPC conversation requires its own thread during the lifetime of the conversation. The RPC Server for CICS ECI can adjust the number of worker threads to the number of parallel requests. The RPC server provides two worker models:

- `FIXED`
  The *fixed* model creates a fixed number of worker threads. The number of worker threads does not increase or decrease during the lifetime of an RPC server instance.
- `DYNAMIC`
  The *dynamic* model creates worker threads depending on the incoming load of RPC requests.

For configuration and technical details, see property `entirex.server.fixedservers` under *Administering the RPC Server for CICS ECI*.

# 3 Administering the RPC Server for CICS ECI

The EntireX RPC Server for CICS® ECI allows standard RPC clients to communicate with CICS programs running on IBM CICS®. It supports the programming languages COBOL and PL/I and transforms the RPC requests from clients into messages to CICS ECI.

## Customizing the RPC Server

The following are used to set up the RPC Server for CICS ECI:

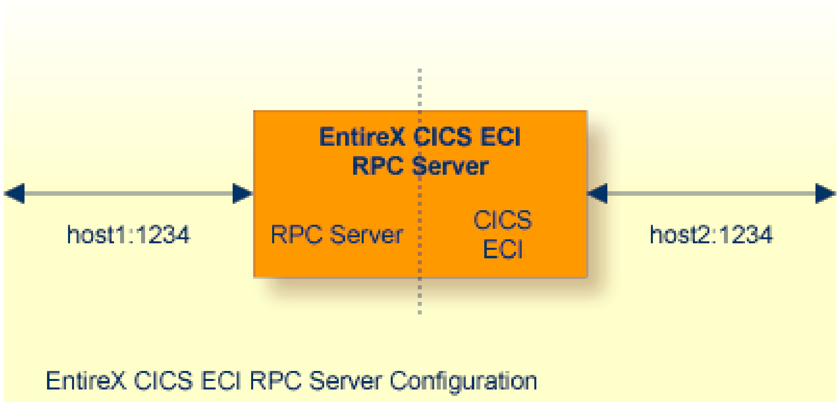
- Configuration File
- Start Script

### Configuration File

The default name of the configuration file is *entirex.cicseci.properties*. The RPC Server for CICS ECI searches for this file in the current working directory.

You can set the name of the configuration file with `-Dentirex.server.properties=<your file name>` with "/" as file separator.

The configuration file contains the configuration for both parts of the RPC Server for CICS ECI.



**Configuring more than one RPC Server**

If you configure more than one RPC Server for CICS ECI that connect to the same broker, the following items must be distinct:

- the trace output file (property `entirex.server.logfile`)
- the log for the Windows Service (property `entirex.server.serverlog`)

### Start Script

The start script for the RPC Server for CICS ECI is called *cicseciserver.bsh* (UNIX) or *cicseciserver.bat* (Windows) and is provided in the *bin* folder of the installation directory. You may customize this file. The RPC Server for CICS ECI itself is contained in the file *entirex.jar*.

## Configuring the RPC Server Side

The RPC Server for CICS ECI uses the properties that start with "entirex.server" for configuring the RPC server side.

Alternatively to the properties, you can use the command-line options. These have a higher priority than the properties set as Java system properties, and these have higher priority than the properties in the configuration file.

| Property Name | Command-line Option | Default | Explanation |
|---|---|---|---|
| entirex.server.brokerid | -broker | localhost | Broker ID. |
| entirex.server.serveraddress | -server | RPC/SRV1/CALLNAT | Server address. |
| entirex.server.userid | -user | CICSECIRPCServer | The user ID for access to the broker. |
| entirex.server.fixedservers | | no | NO The number of worker threads balances between what is specified in entirex.server.minservers and what is specified in entirex.server.maxservers. This is done by a so-called attach thread. At startup, the number of worker threads is the number specified in entirex.server.minservers. A new worker thread starts if the broker has more requests than there are worker threads waiting. If more than the number specified in entirex.server.minservers are waiting for requests, a worker thread stops if its receive call times out. The timeout period is configured with entirex.server.waitserver.<br>See worker model DYNAMIC.<br>YES The number of worker threads specified in entirex.server.minservers is started and the server can process this number of parallel requests.<br>See worker model FIXED. |
| entirex.server.minservers | | 1 | Minimum number of server threads. |

| Property Name | Command-line Option | Default | Explanation |
|---|---|---|---|
| entirex.server.maxservers | | 32 | Maximum number of server threads. |
| entirex.server.restartcycles | -restartcycles | 15 | Number of restart attempts if the Broker is not available. This can be used to keep the RPC Server for CICS ECI running while the Broker is down for a short time. |
| entirex.server.password | -password | yes | The password for secured access to the broker. The password is encrypted and written to the property entirex.server.password.e.<br>■ To change the password, set the new password in the properties file.<br>■ To disable password encryption, set entirex.server.passwordencrypt=no. Default=yes. |
| entirex.server.security | -security | no | Valid values:<br>no \| yes \| auto \| name of BrokerSecurity object. |
| entirex.server.compresslevel | -compresslevel | 0 (no compression) | Enter the text or the numeric value:<br>BEST_COMPRESSION 9<br>BEST_SPEED 1<br>DEFAULT_COMPRESSION (*mapped to* -1 6)<br>DEFLATED 8<br>NO_COMPRESSION 0<br>N 0<br>Y 8 |
| entirex.server.waitattach | | 600S | Wait timeout for the attach server thread. |
| entirex.server.waitserver | | 300S | Wait timeout for the worker threads. |
| entirex.timeout | | 20 | TCP/IP transport timeout. |
| | -help | | Display usage of the command-line parameters. |
| entirex.server.logfile | -logfile | standard output | Name of the log file. |

| Property Name | Command-line Option | Default | Explanation |
|---|---|---|---|
| entirex.trace | -trace | 0 | Trace level.<br>0 No tracing, default.<br>1 Trace all broker calls and other major actions.<br>2 Dump the send and receive buffer.<br>3 Dump the buffers sent to the broker and received from the broker. |

## Configuring the CICS ECI Side

These properties are used to configure the connection to CICS ECI.

Alternatively, you can use the command-line options. These have a higher priority than the properties set as Java system properties, and these have higher priority than the properties in the configuration file.

| Name | Default Value | Explanation |
|---|---|---|
| `cics.host` | | Host name of CICS ECI. Mandatory. |
| `cics.port` | | Port number of CICS ECI. Mandatory. |
| `cics.transaction` | | Name of the CICS mirror transaction that will receive transactions. Mandatory. |
| `entirex.bridge.targetencoding` | `cp037` | Specify the appropriate EBCDIC encoding used by your CICS ECI. This codepage is also used when communicating with the EntireX Broker.<br>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*.<br>Default "`cp037`" is EBCDIC codepage with full Latin-1 character set. |
| `cics.sockettimeout` | `10000` | Socket timeout for connection to CICS ECI (in milliseconds). |
| `cics.userid` | | RACF user ID. Maximum 8 bytes (optional). |
| `cics.password` | | RACF password/PassTicket. Maximum 8 bytes (optional). |
| `cics.application.name` | | Optional. Required if PassTicket is to be used instead of a password. Application name (1-8 characters) as defined in your RACF system. This property is ignored if `cics.password` is set. |
| `cics.secured.signonkey` | | Optional. Required if PassTicket is to be used instead of a password. Secured signon key as defined in your RACF system. Must be exactly 16 characters long. The secured signon key is encrypted and written to the property `cics.secured.signonkey.e`.<br>■ To change the secured signon key, set the new secured signon key in the properties file.<br>■ To disable password and/or secured signon key encryption, set |

| Name | Default Value | Explanation |
|---|---|---|
| | | `entirex.server.passwordencrypt=no`<br>Default=`yes`.<br>This property is ignored if `cics.password` is set. |
| `cics.sslparams` | | SSL parameters (optional). Same syntax as Broker ID. |
| `cics.mapping.folder` | | This is the server-side mapping container, the folder where the RPC server expects server-side mapping files (Designer files with extension .svm).<br>■ If *no* server requires server-side mapping (.svm) or you use server mapping files with extension .cvm only, you can omit this property.<br>■ If *one* server requires server-side mapping (.svm), this property must be specified.<br>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. |

## Using SSL/TLS with the RPC Server

To use SSL with the RPC Server for CICS ECI, you need to configure two sides:

- **CICS ECI Side**
  See parameter `cics.sslparams` under ***Configuring the CICS ECI Side***.
- **RPC Server Side**
  RPC servers 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 servers are always SSL clients. The SSL server can be either the EntireX Broker or Broker SSL Agent. For an introduction see *SSL/TLS, HTTP(S), and Certificates with EntireX* in the platform-independent Administration documentation.

### To use SSL

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

   Use the *URL-style Broker ID* with protocol `ssl://` for the Broker ID. If no port number is specified, port 1958 is used as default. Example:

   ```
   ssl://localhost:22101?trust_store=C:\SoftwareAG\EntireX\etc\ExxCACert.jks&verify_server=no
   ```

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

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

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

   SSL parameters are separated by ampersand (&). See also *SSL/TLS Parameters for SSL Clients*.
3. Make sure the SSL server to which the RPC side connects is prepared for SSL connections as well. The SSL server can be EntireX Broker or Broker SSL Agent. See:
   - *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation
   - Broker SSL Agent in the platform-specific Administration documentation

## Starting the RPC Server

### To start the RPC Server for CICS ECI

- Use the ***Start Script***.

  Or:

  Under Windows you can use the RPC Server for CICS ECI as a Windows Service. See ***Running an EntireX RPC Server as a Windows Service***.

## Stopping the RPC Server

### To stop the RPC Server for CICS ECI

- Use the command `stopService`. See *Stop Running Services* in Command Central's Command-line Interface.

  Or:

  Stop the service using Command Central's Graphical User Interface. See *Stopping a Service*.

  Or:

  Use the command-line utility `etbcmd`. See `ETBCMD` under *Broker Command-line Utilities* in the platform-specific Administration documentation.

  Or:

  Use `CTRL-C` in the session where you started the RPC server instance.

  Or:

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

## Pinging the RPC Server

### To ping the RPC Server for CICS ECI

- Enter the following command:

  ```
  java -classpath "$EXXDIR/classes/entirex.jar" ↩
  com.softwareag.entirex.rpcping.RPCServerPing -p <admin_port>
  ```

  where `admin_port` is the number of the administration port.

  The `ping` command returns "`0`" if the server is reachable, and "`1`" if the server cannot be accessed.

## Running an EntireX RPC Server as a Windows Service

For general information see *Running an EntireX RPC Server as a Windows Service* in the Windows Administration documentation.

### To run the RPC Server for CICS ECI as a Windows Service

1. Customize the ***Start Script*** according to your system installation.

   **Note:** The script must pass external parameters to the RPC server and use the reduced signaling of the JVM (option `-Xrs`):

   ```
   java -Xrs com.softwareag.entirex.cics.bridge.CicsRPCServer %*
   ```

   If `-Xrs` is not used, the JVM stops and an entry 10164002 is written to the event log when the user logs off from Windows.

   See also ***Starting the RPC Server***.

2. Test your RPC server to see whether it will start if you run your script file.
3. Use the *EntireX RPC Service Tool* and install the `RPCService` with some meaningful extension, for example `MyServer`. If your ***Start Script*** is *cicseciserver.bat*, the command will be

```
RPCService -install -ext MyServer ↩
-script install_path\EntireX\bin\cicseciserver.bat
```

The log file will be called *RPCservice_MyServer.log*.

4 In **Windows Services** menu (**Control Panel** > **Administrative Tools** > **Services**) select the service: `Software AG EntireX RPC Service [MyServer]` and change the property `Startup Type` from "Manual" to "Automatic".

## Application Identification

The application identification is sent from the RPC Server for CICS ECI to the Broker. It is visible with Broker Command and Information Services.

The identification consists of four parts: name, node, type, and version. These four parts are sent with each Broker call and are visible in the trace information.

For the RPC Server for CICS ECI, these values are:

| Identification Part | Value |
|---|---|
| Application name | `ANAME=RPC Server for CICS ECI` |
| Node name | `ANODE=<`*`host name`*`>` |
| Application type | `ATYPE=Java` |
| Version | `AVERS=10.8.0.0` |

# 4 Preparing IBM CICS for ECI

This chapter describes how to set up the External Call Interface (ECI) within CICS.

⚠ **Important:** If the terms and concepts in this chapter are unfamiliar to you, ask an appropriate CICS system programmer. Only authorized personnel should make changes to mainframe computer systems.

## Defining an ECI Service

The `DFH$SOT` group contains three TCP/IP services. For our purposes, the pertinent service is ECI. The ECI service has, defined in it, the TCP/IP port number through which the CICS region listens to the ECI. By default, IBM predefines the TCP/IP port number as "1435". If this port is already reserved for another CICS region, you may have to define a different port number in the ECI service. For more information, see your IBM documentation.

You must specify `SOCKETCLOSE(NO)` on the `TCPIPSERVICE`. If you need to specify a timeout for a task initiated using ECI over TCP/IP, specify an `RTIMOUT` value on the mirror transaction. Note that the standard mirror, `CPMI`, is defined with profile `DFHCICSA`, which specifies `RTIMOUT(NO)`. This means that long running mirrors will wait indefinitely for data, unless you specify a different `RTIMOUT` for the mirror transaction.

### To define an ECI service without security

1. Use `CEDA ALTER` to supply the `ECI TCPIPService` with a unique TCP/IP port number.
2. Use `CEDA ALTER` to set the `ECI ATTACHSEC` to "`LOCAL`"

### To define an ECI service with security

1. Copy the ECI `TCPIPService` to an ECIS `TCPIPSERVICE`.
2. Use `CEDA ALTER` to supply the ECIS `TCPIPSERVICE` with a unique TCP/IP port number.
3. Use `CEDA ALTER` to set the `ECIS ATTACHSEC` to "`VERIFY`".

## Installation Verification

A successful installation should pass the following verification tests:

1. `CEMT INQUIRE TCPIPSERVICE(*)` should now display your services with status OPE. A status of CLO might indicate an already used (not unique) TCP/IP port number.
2. `CEMT I TCPIPSERVICE(ECI)` to verify `TCPIPSERVICE` in CICS.
3. `CEMT I TRA(CIEP)` to verify transaction `CIEP` in CICS.

4. `CEMT I PROGRAM(DFHIEP)` to verify program `DFHIEP` in CICS.
5. `CEMT I TRAN(CPMI)` to verify transaction `CPMI` in CICS.
6. `CEMT I PROGRAM(DFHMIRS)` to verify program `DFHMIRS` in CICS.
7. `CEMT I TD(CIEO)` to verify program `TQ` queue `CIEO`.
8. Verify that the groups `DFHISC` `DFHDCTG` and `DFHIPECI` are added to the active autoinstall.

**Tip:** You can check the mentioned programs and transactions and TD queues by using `CEMT INQUIRE TCPIPSERVICE(*)`.

## Error Handling

This table describes the handling of errors in the CICS ECI connection or the RPC Server for CICS ECI.

| Problem | Handling |
|---|---|
| A CICS program sends abend code in response. | The CICS session is closed and the next call opens a different session. |
| The TCP/IP connection is lost with a `SocketTimeoutException`. | The CICS session is closed and the next call opens a different session. |
| The TCP/IP connection is lost with an `EOFException`. | ■ The TCP/IP socket is closed and the next call opens a different session.<br>■ There are no further attempts to send bytes on the TCP/IP connection that received the `EOFException`. |

# 5 Server-side Mapping Files

Server-side mapping files have the extension .svm.

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.

See also *Source Control of Server Mapping Files* | *Comparing Server Mapping Files* | *When is a Server Mapping File Required?* | *Migrating Server Mapping Files* in the Designer documentation.

## Server-side Mapping Files in the RPC Server

Server-side mapping files (.svm) are stored as ordinary files in the server-side mapping container folder.

- If *no* server requires server-side mapping (.svm) or you are using server mapping files with extension .cvm only, you can omit the property `cics.mapping.folder`.
- If *one* server requires a server-side mapping file (.svm), provide the property `cics.mapping.folder`. See ***Configuring the CICS ECI Side***.

## Undeploying Server-side Mapping Files from the RPC Server

Undeploy a server mapping file (Designer file with extension .svm) from the RPC server manually. See *Server Mapping Files for COBOL*.

### To undeploy a server-side mapping file manually

- Delete the server-side mapping file from the server-side mapping container (directory or folder). See *Server Mapping Files for COBOL*.

## Change Management of Server-side Mapping Files

Under UNIX and Windows, change management for a directory or folder (server-side mapping container, see ***Server-side Mapping Files in the RPC Server***) is similar to change management within ordinary operating system directories (folders). All updates to the directory or folder done after a backup must be kept.

All Designer server-side mapping files (.svm) added since the last backup should be available. See *Server Mapping Files for COBOL* in the Designer documentation.

## List Deployed Server-side Mapping Files

Use the Windows Explorer or the UNIX `ls` command to list the contents of the server-side mapping container (directory or folder). See ***Server-side Mapping Files in the RPC Server***.

## Check if a Server-side Mapping File Revision has been Deployed

Server-side mapping files in the server-side mapping container correspond to Designer file with extension .svm (same format). See *Server Mapping Files for COBOL* in the Designer documentation. Each line relates to an IDL program and contains a creation timestamp at offset 276 (decimal) in the format *`YYYYMMDDHHIISST`*. Precision is 1/10 of a second. The creation timestamp can be checked.

The timestamp can be found on the same offset in the server-side mapping files stored in the server-side mapping container (directory or folder). See ***Server-side Mapping Files in the RPC Server***.

# 6 Scenarios

# COBOL Scenarios

## Scenario I: Calling an Existing COBOL Server

### To call an existing COBOL server

1. Use the IDL Extractor for COBOL to extract the Software AG IDL and, depending on the complexity, also a server mapping file. See *When is a Server Mapping File Required?* under *Server Mapping Files for COBOL* in the Designer documentation.
2. Build an EntireX RPC client using any EntireX wrapper. For a quick test you can:
   - use the IDL Tester; see *EntireX IDL Tester* in the Designer documentation
   - generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester* in the XML/SOAP Wrapper documentation

See *Basic RPC Server Examples -* `CALC`*,* `SQUARE` for COBOL RPC Server examples.

## Scenario II: Writing a New COBOL Server

### To write a new COBOL server

1. Use the COBOL Wrapper to generate a COBOL server skeleton and, depending on the complexity, also a server mapping file. See *When is a Server Mapping File Required?* under *Server Mapping Files for COBOL* in the Designer documentation. Write your COBOL server and proceed as described under *Using the COBOL Wrapper for the Server Side*.
2. Build an EntireX RPC client using any EntireX wrapper. For a quick test you can:
   - use the IDL Tester; see *EntireX IDL Tester* in the Designer documentation
   - generate an XML mapping file (XMM) and use the XML Tester for verification; see *EntireX XML Tester* in the XML/SOAP Wrapper documentation

See *Basic RPC Server Examples -* `CALC`*,* `SQUARE` in the COBOL Wrapper documentation for COBOL RPC Server examples.