Administering the Micro Focus RPC Server

The EntireX Micro Focus COBOL RPC Server allows standard RPC clients to communicate with COBOL servers written with Micro Focus COBOL. It works together with the *COBOL Wrapper* and the *IDL Extractor for COBOL*.

This chapter covers the following topics:

- Customizing the RPC Server
- Configuring the RPC Server
- Locating and Calling the Target Server
- Using SSL or TLS with the RPC Server
- Starting the RPC Server
- Stopping the RPC Server
- Activating Tracing for the RPC Server

Customizing the RPC Server

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

- Micro Focus COBOL Runtime
- Configuration File
- Start Script

Micro Focus COBOL Runtime

The COBOL runtime, for example *Micro Focus Server*, has to be installed according to the Micro Focus documentation. It is not delivered with this package. Provide the location of the COBOL runtime in the *Start Script*.

If a COBOL runtime is not provided, the Micro Focus RPC Server cannot be started and an error message is given.

Configuration File

The name of the delivered example configuration file is *microfocusserver.cfg*. The configuration file contains the configuration for the Micro Focus RPC Server. The following settings are important:

- connection information such as broker ID, server address (class, name, service)
- location and usage of server-side mapping container; see *Usage of Server Mapping Files*.

- scalability parameters
- trace settings
- etc.

For more information see *Configuring the RPC Server*.

Start Script

The name of the start script is platform-dependent:

- UNIX: *microfocusserver.bsh*
- Windows: *microfocusserver.bat*

The start script for the Micro Focus RPC Server contains the following:

- the location of the Micro Focus COBOL runtime
- paths to the called COBOL server; see *Configuration Approaches*
- the configuration file used; see *Configuration File*
- etc.

Configuring the RPC Server

The following rules apply:

- In the configuration file:
  - Comments must be on a separate line.
  - Comment lines can begin with '*', '/' and ';'.
  - Empty lines are ignored.
  - Headings in square brackets [<topic>] are ignored.
  - Keywords are not case-sensitive.
- Underscored letters in a parameter indicate the minimum number of letters that can be used for an abbreviated command.

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

Parameter	Default	Values	Req/ Opt
brokerid	localhost	Broker ID used by the server. See *Using the Broker ID in Applications*. Example: `brokerid=myhost.com:1971`	R
class	RPC	Server class part of the server address used by the server. The server address must be defined as a service in the broker attribute file (see *Service-specific Attributes* under *Broker Attributes*). Case-sensitive, up to 32 characters. Corresponds to `CLASS`. Example: `class=MyRPC`	R
codepage		Depending on the internationalization approach, the codepage (locale string) where incoming data is provided to the COBOL server. Conversely, the COBOL server must provide outgoing data in the given codepage, otherwise unpredictable results occur. See *What is the Best Internationalization Approach to use?* under *Internationalization with EntireX* for information on which internationalization approach requires a codepage (locale string). By default, no codepage is transferred to the broker. For the most popular internationalization approach, *ICU Conversion*, the correct codepage (locale string) must be provided. This means it must: • follow the rules described under *Locale String Mapping* • be a codepage supported by the broker • be the codepage used in your environment for file and terminal IO, otherwise unpredictable results may occur. Example: `codepage=iso-8859-1`	R (UNIX) O (Windows)

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

deployment	NO	Activates the deployment service, see *Deployment Service*. Required to use the Server Mapping Deployment Wizard. See *Server Mapping Deployment Wizard* in the EntireX Workbench documentation. **YES** Activates the deployment service. The RPC server registers the deployment service in the broker. **NO** The deployment service is deactivated. The RPC server does not register the deployment service in the broker. Example: `deployment=yes`	O

Parameter	Default	Values	Req/ Opt
encryptionlevel	0	Enforce encryption when data is transferred between client and server. Requires EntireX Security. See ENCRYPTION-LEVEL under *Broker ACI Fields*. **0** Encryption is enforced. **1** Encryption is enforced between server and broker kernel. **2** Encryption is enforced between server and broker kernel, and also between client and broker. Example: encryptionlevel=2	O
logon	YES	Execute broker functions LOGON/LOGOFF in worker threads. Must match the setting of the broker attribute AUTOLOGON. Reliable RPC requires logon set to YES. See *Reliable RPC*. **NO** No logon/logoff functions are executed. **YES** Logon/logoff functions are executed. Example: logon=no	O
marshalling	COBOL	The Micro Focus RPC Server supports COBOL. See also *Locating and Calling the Target Server*. Marshalling=(LANGUAGE=COBOL, flavor=MF) must be provided. Do not change these settings. The COBOL servers are called directly without a server interface object. So-called server mapping files are used to call the COBOL server correctly if one is available. See *Usage of Server Mapping Files*.	O
password	no default	Password for broker logon. Case-sensitive, up to 32 characters. For more information see broker ACI control block field PASSWORD. Example: password=MyPwd	O
restartcycles	15	Number of restart attempts if the broker is not available. This can be used to keep the Micro Focus RPC Server running while the broker is down for a short time. A restart cycle will be repeated at an interval which is calculated as follows: timeout + ETB_TIMEOUT + 60 seconds where timeout is the RPC server parameter (see this table), and ETB_TIMEOUT is the environment variable (see *Environment Variables in EntireX*) When the number of cycles is reached and a connection to the broker is not possible, the RPC server stops. Example: restartcycles=30	O
servername	SRV1	Server name part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes* under *Broker Attributes*. Case-sensitive, up to 32 characters. Corresponds to SERVER of the broker attribute file. Example: servername=mySrv	R
service	CALLNAT	Service part of the server address used by the server. The server address must be defined as a service in the broker attribute file. See *Service-specific Attributes* under *Broker Attributes*. Case-sensitive, up to 32 characters. Corresponds to SERVICE attribute of the broker attribute file. Example: service=MYSERVICE	R

Parameter	Default	Values	Req/ Opt
smhport	0	The port where the server listens for commands from the System Management Hub (SMH). If this port is 0 (default), no port is used and management by the SMH is disabled. Example: `smhport=3001`	O
ssl_file	no default	Set the SSL parameters. See *Using SSL or TLS with the RPC Server* for examples and more information.	O

svm		Usage and anchor of the server-side mapping container (directory or folder). See *Server-side Mapping Files in the RPC Server*. The RPC server needs write access to the server-side mapping container. There are also client-side mapping files that do not require configuration here. See *Server Mapping Files for COBOL*
		
		`SVM=(PATH=`*`path`*`)`
		
		path The path to the anchor of the server-side mapping container.
		
		Example for UNIX:
		`SVM=(PATH=../config/svm)`
		
		Example for Windows:
		`SVM=(PATH=..\config\svm)`
		
		See also *Usage of Server Mapping Files*.
			O

timeout	60	Timeout in seconds, used by the server to wait for broker requests. See broker ACI control block field `WAIT` for more information. Also influences `restartcycles`. Example: `timeout=300`	O
tracedestination	`ERXTrace.`*`nnn`*`.log`	The name of the destination file for trace output. By default the main trace file name is ERXTrace.*nnn*.log, where *nnn* can be in the range from 001 to 005. **UNIX** The trace file is located in the current working directory. **Windows** The trace file is located in a subfolder of the windows folder My Documents. Example: `tracedestination=ERXTrace.log`	O

tracelevel	`None`	Trace level for the server. See also *Activating Tracing for the RPC Server*.
		
		`tracelevel = None | Standard | Advanced | Support`
		
		`None` No trace output.
		`Standard` For minimal trace output.
		`Advanced` For detailed trace output.
		`Support` This trace level is for support diagnostics and should only be switched on when requested by Software AG support.
		
		Example:
		`tracelevel=standard`
			O

Parameter	Default	Values	Req/ Opt
traceoption	None	Additional trace option if trace is active. **None** No additional trace options. **STUBLOG** If tracelevel is Advanced or Support, the trace additionally activates the broker stub log. **NOTRUNC** Normally if a data buffer larger than 8 KB is traced, the buffer trace is truncated. Set this option to write the full amount of data without truncation. **Note:** This can increase the amount of trace output data dramatically if you transfer large data buffers. Example: traceoption=(STUBLOG,NOTRUNC)	O
userid	ERX-SRV	Used to identify the server to the broker. See broker ACI control block field USER-ID. Case-sensitive, up to 32 characters. Example: userid=MyUid	R

Parameter	Default	Values	Req/ Opt

workermodel	SCALE,1,3,slowshrink	The Micro Focus RPC Server can be configured to
		• adjust the number of worker threads to the current number of client requests:
		`workermodel=(SCALE,from,thru [,slowshrink | fastshrink] [,noisolation | isolation])`
		• use a fixed number of worker threads:
		`workermodel=(FIXED,number [,noisolation | isolation])`
		FIXED A fixed *number* of worker threads is used by the Micro Focus RPC Server.
		SCALE The number of worker threads is adjusted to the current number of client requests. With the *from* value, the minimum number of active worker threads can be set. The *thru* value restricts the maximum number of worker threads.
		slowshrink The RPC server stops all worker threads not used in the time specified by the timeout parameter, except for the number of workers specified as minimum value. This is the default if SCALE is used.
		fastshrink The RPC server stops worker threads immediately as soon as it has finished its conversation, except for the number of workers specified as minimum value.
		noisolation Calls to the COBOL server are executed within the Micro Focus RPC Server. If the COBOL server causes a COBOL runtime error, the Micro Focus RPC Server stops.
		isolation Default. Calls to the COBOL server are executed in separate processes. If the COBOL server causes a COBOL runtime error, the Micro Focus RPC Server does not stop and continues.
		Example:
		`workermodel=(SCALE,2,5)`
			O

Locating and Calling the Target Server

Introduction

The Micro Focus RPC Server is able to call standard libraries (Windows DLLs or UNIX so|sl); Micro Focus proprietary formats such as intermediate code (*.int); generated code (*.gnt); and intermediate or generated code packaged in libraries (*.lbr). See the following table:

Executable Format	File Extension	File Name	Entry Point	Notes	Configuration
Operating system standard library with multiple server	.so|sl (UNIX) or .dll (Windows)	IDL library	IDL program	1,2	1
Operating system standard library with single server	.so|sl (UNIX) or .dll (Windows)	IDL program	IDL program	1,3,4	2
Micro Focus proprietary intermediate code	.int	IDL program		4	2
Micro Focus proprietary generated code	.gnt	IDL program		4	2
Micro Focus proprietary library with multiple server	.lbr	IDL library	IDL program	2,5	2
Micro Focus proprietary library with single server	.lbr	IDL program	IDL program	3,4,5	2

Notes

1. This type of library is a standard library (UNIX shared library or Windows DLL).

2. This type of library may contain multiple COBOL servers. The IDL library name is used to form the operating system file name. The COBOL server names (entry points) are taken as follows:

   - if the COBOL Wrapper is used, by default from the IDL program names. The IDL program name can be different if it is renamed during the wrapping process, see *Customize Automatically Generated Server Names*

   - if the IDL Extractor for COBOL is used, from the COBOL program IDs. The IDL program name can be different if it is renamed during the extraction process in the *COBOL Mapping Editor*

   If the IDL program name is different, a server mapping is required, See *Usage of Server Mapping Files*.

3. This type of library must contain one COBOL server only.

4. The IDL library name is not used. The COBOL server name (operating system file name and its entry point) are taken as follows:

- if the COBOL Wrapper is used, by default from the IDL program name. The IDL program name can be different if it is renamed during the wrapping process, see *Customize Automatically Generated Server Names*
- if the IDL Extractor for COBOL is used, from the COBOL program ID. The IDL program name can be different if it is renamed during the extraction process in the *COBOL Mapping Editor*

If the IDL program name is different, a server mapping is required, See *Usage of Server Mapping Files*.

5. Intermediate (*.int) or generated (*.gnt) code must be packaged in the library.

Configuration Approaches

There are two approaches to access the COBOL server during runtime, which depend on the executable format (see table above):

1. The operating system's standard call mechanism is used to call libraries. Make sure your server(s) are accessible, for example:
   - under UNIX with the `LD_LIBRARY_PATH` environment variable
   - under Windows with the `PATH` environment variable
2. The Micro Focus environment variable `COBPATH` must be set before starting the RPC server. It lists all paths where a search for COBOL servers is to be performed. See the Micro Focus documentation for more information.

For both approaches, the start script of the Micro Focus RPC Server is an appropriate place to set the environment variables. See *Start Script*.

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

Using SSL or TLS with the RPC Server

There are two ways of specifying SSL or TLS, depending on the complexity of the parameters:

- as part of the Broker ID for short parameters, the simplest way
- using the SSL file, a text file containing more complex parameters.

For more information, see *SSL or TLS and Certificates with EntireX*.

Specifying the SSL or TLS Parameters as Part of the Broker ID

The simplest way to specify SSL or TLS parameters is to add them to the Broker ID.

Example:

```
ssl://ETB001?TRUSTSTORE=whatever
```

Specifying the SSL or TLS Parameters in a Separate File

For complex SSL or TLS parameters there is the SSL file, a text file containing the parameters.

The `SSL_FILE` keyword points to this text file.

To specify the SSL or TLS parameters in the SSL file

1. Set the parameters as described under *Running Broker with SSL or TLS Transport* under z/OS | UNIX | Windows.
2. Prefix/suffix the Broker ID with the SSL key.

   Example:

   ```
   brokerid=SSL://ETB001
   .
   .
   ssl_file=C:\mySSLdirectory\mySSLParms.txt
   ```

Starting the RPC Server

To start the Micro Focus RPC Server

- Use the script *microfocusrpcserver* in the folder *bin* to start the Micro Focus RPC Server. You may customize this file.

  Or:
  Use the RPC server agent in the System Management Hub to configure and start the Micro Focus RPC Server.

  See *Administering the EntireX RPC Servers using System Management Hub* under UNIX | Windows for details.

  Or:
  Use the Micro Focus RPC Server as a Windows service, see *Running an EntireX RPC Server as a Windows Service*.

Stopping the RPC Server

To stop the Micro Focus RPC Server

- Use the RPC server agent in the SMH to stop the Micro Focus RPC Server.

  Or:
  Use the agent for the broker. Use `Deregister` on the service, specified with the parameters class/servername/service.

Activating Tracing for the RPC Server

To switch on tracing for the RPC server

- Set the parameters `tracelevel` and `tracedestination`. See *Configuring the RPC Server*.

  To evaluate the return codes, see *Error Messages and Codes*. See also *Tracing the RPC Server* under UNIX | Windows.