webMethods EntireX

Designer

Version 10.3

October 2018

WEBMETHODS

This document applies to webMethods EntireX Version 10.3 and all subsequent releases.

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

Copyright © 1997-2018 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at http://softwareag.com/licenses.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at http://softwareag.com/licenses/ and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at http://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: EXX-EEXXWORKBENCH-103-20191129

Table of Contents

Software AG Designer v
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
2 Scope of the Designer 5
Editors 6
EntireX Wrappers 7
Software AG IDL Extractors 9
Other Components 10
3 Supported File Types 11
Input Files 12
Output Files 13
4 Server Mapping Files for Natural 15
Server Mapping Files in the Designer 16
When is a Server Mapping File Required? 18
Source Control of Server Mapping Files 18
Comparing Server Mapping Files 19
5 Server Mapping Files for COBOL 21
Server Mapping Files in the Designer 22
How to Set the Type of Server Mapping Files 25
When is a Server Mapping File Required? 26
Migrating Server Mapping Files 31
Source Control of Server Mapping Files 32
Comparing Server Mapping Files 32
6 Server Mapping Deployment Wizard 33
Introduction 35
Using 35
Preferences 39
Command-line Mode 42
7 EntireX IDL Tester 45
Calling the IDL Tester 46
Using the Broker and RPC User ID/Password 48
8 Using EntireX Custom Wrappers 51
Define EntireX Custom Wrappers 52
Running a Custom Wrapper 59
9 Using EntireX in the Designer Command-line Mode 61
Command Line under Windows 62
Command Line under Linux 63
List of all Commands 63
10 Storing IDL Properties in an External File 67

Software AG Designer

The EntireX design-time user interface is implemented as part of Software AG Designer and delivered as an Eclipse feature. It provides various wrappers, IDL extractors, editors and testers for generating and testing RPC communication between RPC servers and RPC client applications.

Scope	Lists the EntireX components of the Designer, with links to more information.
Supported File Types	EntireX file types supported by Designer.
Server Mapping Files for Natural	Handling server mapping files for Natural in the Designer. Server mapping files are used at runtime to marshal and unmarshal the RPC data stream. This document provides information on source control, comparing etc. of server mapping files.
Server Mapping Files for COBOL	Handling server mapping files for COBOL in the Designer. Server mapping files are used at runtime to marshal and unmarshal the RPC data stream. This document provides information on migration, source control, comparing etc. of server mapping files.
Deployment Wizard	How to deploy server-side mapping files (Designer files with extension .svm) with the Server Mapping Deployment Wizard.
IDL Tester	Using the EntireX IDL Tester.
Custom Wrappers	How to define your own template-based wrappers.
Command-line Mode	Using EntireX design-time features from a Designer command line.
Storing IDL Properties	Using an external file to store IDL properties for export/import.

See also *Installing EntireX Design-time outside the Designer*. This document is applicable if you want to install the EntireX design-time separately, that is, not as part of the full webMethods EntireX installation.

If you are not yet familiar with Eclipse, see the Eclipse online help at *http://www.eclipse.org/documentation/*, or start the Designer and then choose **Help** > **Help Contents**. General information on Eclipse can then be found under *Workbench User Guide*. When working with the Designer, the online help also provides help for the currently installed Software AG products; this can be found under *Software AG Designer Guides*.

You can find an introduction to the Designer on the Software AG Documentation website at **http://documentation.softwareag.com** under *Guides for Shared Tools* > *Working with Software AG Designer*.

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

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

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **http://documentation.softwareag.com**. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.asp** and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at **http://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

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 Scope of the Designer

- Editors 6
- EntireX Wrappers 7
- Software AG IDL Extractors 9
- Other Components 10

Editors

IDL Editor

A Software AG IDL file contains definitions of the interface between client and server. The IDL file is used by Software AG wrappers to generate RPC clients, RPC servers and tester etc. on the basis of these definitions. The IDL file can be edited by the IDL Editor provided by plug-ins for Eclipse.

Merging and Refactoring Software AG IDL

IDL refactoring is a process that checks all programs and structures in a single library if they contain identical groups. All identical groups are extracted in a single structure in the same library, and replaced with a structure reference. If a structure exists that is identical to the structure to be created, all references will point to the existing structure and a new one will not be created. Two groups are identical if each group has the same number and order of parameters, and each parameter in one group has the same name and the same type as the corresponding parameter in the other group. IDL refactoring can be performed on single or multiple IDL files.

XML Mapping Editor

The EntireX XML Mapping Editor allows you to map XML document structures to IDL libraries, programs and parameters. The mappings can be defined for the request and response to the server application, or from the server to the client. The input for the XML Mapping Editor can be a Software AG IDL file and/or an IDL-XML mapping file (perhaps produced by a previous XML Mapping Editor session or by importing a WSDL file, XML Document or XML Schema). The output is an IDL-XML mapping file, other XML structure definitions (such as sample XML files), and perhaps a created or changed IDL file.

EntireX Wrappers

In EntireX terms, a wrapper is a tool contained in the Designer to generate interface objects based on a Software AG IDL file and additional wrapping properties. The following wrappers are provided:

C Wrapper
EntireX C Wrapper provides access to RPC-based components from C applications. It enables you to develop both client and server applications.

COBOL Wrapper
EntireX COBOL Wrapper provides access to RPC-based components from COBOL applications. It enables you to develop both client and server applications.

Custom Wrappers
The EntireX Custom Wrappers are user-configurable, template-based wrappers and need a Software AG IDL file, a template (e.g., client or server) and the Software AG IDL Compiler.

DCOM Wrapper
The EntireX DCOM Wrapper generates DCOM-enabled components using RPC technology. This so-called “wrapping” makes it possible to treat existing applications as ActiveX components.

.NET Wrapper
The EntireX .NET Wrapper provides access to RPC servers for .NET client applications and access to .NET servers for any RPC client. The .NET Wrapper generation tools of the Designer take as input a Software AG IDL file, which describes the interface of the RPC, and generates C# classes that implement the methods and data types of the interface.

Integration Server Wrapper
The webMethods Integration Server Wrapper generates Integration Server adapter services and listeners from a Software AG IDL file within an Integration Server connection definition.

Java Wrapper
The EntireX Java Wrapper provides access to EntireX RPC-based components from Java applications. With EntireX Java RPC you can develop both client and server applications written in Java. Java applets can also be used as EntireX RPC clients.

Java Wrapper for Natural
The EntireX Java Wrapper for Natural allows you to generate EntireX Java client interface objects from Natural subprograms in a NaturalONE project in Eclipse. The generated Java client interface objects can be used by Java application developers to access Natural server components, using EntireX/Natural RPC.

Natural Wrapper
The Natural Wrapper allows you to develop Natural client applications that access RPC-based server components, and to create Natural RPC server skeletons you can use as a basis to write a Natural RPC server that can be accessed by RPC clients.

PL/I Wrapper

The EntireX PL/I Wrapper provides access to RPC-based components from PL/I applications. It enables you to develop both client and server applications.

Web Services Wrapper

The EntireX Web Services Wrapper is a wizard that generates and optionally deploys Web services (Designer file with extension .aar) to offer an RPC server - for example a COBOL or Natural RPC server - as a Web service. The generated XML/SOAP mapping file (Designer file with extension .xmm) can also be used to enable RPC clients - for example a COBOL or Natural client - consuming (or calling) a Web service.

Web Services Wrapper for Natural

The EntireX Web Services Wrapper for Natural allows you to develop Web Services that access Natural server components, using EntireX/Natural RPC.

XML/SOAP Wrapper

The EntireX XML/SOAP Wrapper enables XML-based communication to EntireX/Natural RPC servers and communication from EntireX/Natural RPC clients to XML-based servers.

Software AG IDL Extractors

An extractor is a tool contained in the Designer to extract a Software AG IDL file. The following extractors are provided:

IDL Extractor for COBOL
The Software AG IDL Extractor for COBOL enables you to extract the interface of a COBOL server and transforms it into a Software AG IDL and a Software AG server mapping file. Both files are required to provide access for any RPC client to the COBOL server.

IDL Extractor for Integration Server
The Software AG IDL Extractor for webMethods Integration Server is a wizard that reads a package from the Integration Server and generates a Software AG IDL file from all the existing services and nodes. Each service results in a program in the IDL file. All parameters of the services are mapped to an IDL alphanumeric data type, available as variable (AV) or fixed (A*n*) length.

IDL Extractor for Natural
The Software AG IDL Extractor for Natural extracts a Software AG IDL definition from a Natural source in a Natural project in Eclipse, or from an object within a Natural RPC environment.

IDL Extractor for PL/I
The Software AG IDL Extractor for PL/I extracts a Software AG IDL file from a PL/I source. The PL/I source can be located in the file system or accessed remotely within a PL/I RPC environment definition.

IDL Extractor for WSDL
The Software AG IDL Extractor for WSDL is a wizard that generates Web service client artifacts from a WSDL file. With these artifacts, EntireX RPC client applications can access external Web services.

IDL Extractor for XML Document
The Software AG IDL Extractor for XML Document generates an IDL File and a related XML mapping file (XMM) from a given XML document.

IDL Extractor for XML Schema
The Software AG IDL Extractor for XML Schema generates an IDL File and a related XML mapping file (XMM) from given XML schema files.

Other Components

CentraSite Integration

Web services created with EntireX can be published to CentraSite. CentraSite offers enhanced registry functionality, and also repository functionality that enables you to store Web services artifacts and register interdependencies for impact analysis.

Default Broker View

The EntireX Default Broker View is part of the Designer. It displays the status of the EntireX Default Broker and the active RPC Services registered to it.

UDDI Registration

EntireX UDDI Registration is a tool with which you can register a Web service with any UDDI registry for which you have an account.

3 Supported File Types

- Input Files 12
- Output Files 13

Input Files

The following table lists the file types that the Designer can use as input. Some of the file types may have different content. The content determines which files you can generate. The actions in the context menu depend on the file type.

File Type	Content	Generated Files
cvm	Client-side mapping file that completes the related IDL of the same name with server mapping information to successfully call the target RPC server customer program. Contains COBOL mapping (see ***Server Mapping Files for COBOL***) or Natural mapping (see ***Server Mapping Files for Natural***).	
idl	Software AG IDL for RPC, see *Software AG IDL File* in the IDL Editor documentation.	C client, C server, Java client, Java server, Java tester, COBOL client, COBOL server, C# client, DCOM object, XMM file, PL/I client, PL/I server.
svm	Server-side mapping file that completes the related IDL of the same name with server mapping information to successfully call the target RPC server customer program. Contains COBOL mapping; see ***Server Mapping Files for COBOL***.	
wsdl	Web service description file.	IDL file, XMM mapping file.
xml	XML document.	IDL file, XMM mapping file.
xmm	EntireX XMM file with element mapping.	
xmm	EntireX XMM file with attribute mapping.	
xmm	EntireX XMM file with user-defined mapping.	
xmm	EntireX XMM file with SOAP mapping.	WSDL file.
xsd	XML Schema file.	IDL file, XMM mapping file.

Output Files

The following table lists the file types generated by the Designer.

File Type	Content	Generated from
aar	Service archive for Web Services Stack.	IDL file with XML mapping.
cvm	Client-side mapping file (Natural | COBOL) that completes the related IDL of the same name with server mapping information to successfully call the target RPC server customer program.	COBOL source; Natural source.
idl	Software AG IDL for RPC.	Import XSD, import XML, import WSDL.
svm	Server-side mapping file (COBOL) that completes the related IDL of the same name with server mapping information to successfully call the target RPC server customer program. Contains COBOL mapping.	COBOL source.
wsdl	Web service description file.	IDL file.
xml	XML document as test document for the XML Tester.	IDL file.
xmm	EntireX XMM file with element mapping.	IDL file with XML mapping.
xmm	EntireX XMM file with attribute mapping.	IDL file with XML mapping.
xmm	EntireX XMM file with user-defined mapping.	IDL file with XML mapping.
xmm	EntireX XMM file with SOAP mapping.	IDL file with XML mapping.

4 Server Mapping Files for Natural

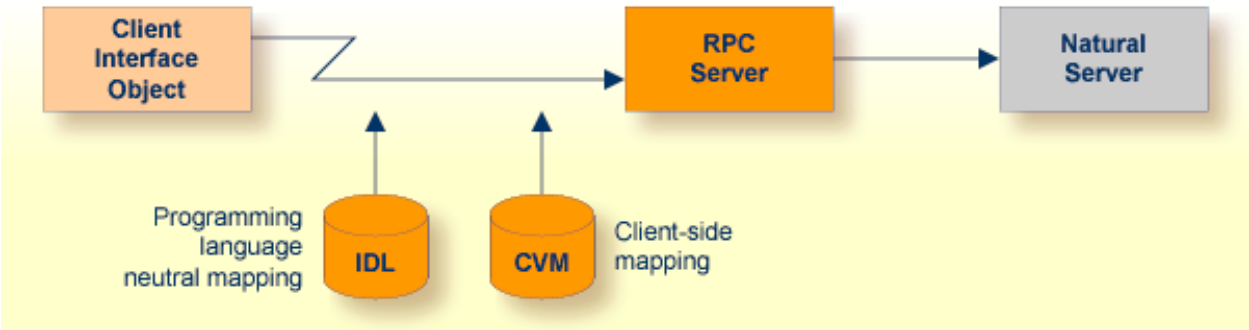
- Server Mapping Files in the Designer 16
- When is a Server Mapping File Required? 18
- Source Control of Server Mapping Files 18
- Comparing Server Mapping Files 19

Server mapping enables the RPC server to correctly support special Natural syntax such as `REDEFINE`s, special Natural Mapping Editor operations, etc. If one of these elements is used the IDL Extractor for Natural automatically extracts a server mapping file in addition to the IDL file (interface definition language). Also the Natural Wrapper may generate a server mapping file for RPC server generation. The server mapping file is used at runtime to marshal and unmarshal the RPC data stream. Client-side mapping files (Designer files with extension .cvm) are used for Natural only.

Server Mapping Files in the Designer

The server mapping file for Natural in the Designer is of type client-side. There are no server-side mapping files as with COBOL.

Client-side Mapping



The following rules apply to a server mapping file for Natural:

- The mapping file is of type client-side. There are no server-side mapping files as with COBOL.
- Mapping files have the extension .cvm.
- The mapping file has to relate to an appropriate IDL file. Always keep the IDL file and the mapping file together in the same folder.
- The mapping file is created if required by the Natural Wrapper | Extractor. See *When is a Server Mapping File Required?*
- If an IDL file has a corresponding mapping file, at least one of the IDL programs in the IDL file requires server-mapping information to correctly call the target server. For those IDL programs, there is a server mapping (corresponding to a line) in the server mapping file.
- If there is an IDL file but no corresponding mapping file, there is no IDL program that requires server mapping information.

When is a Server Mapping File Required?

IDL Extractor for Natural

Natural Syntax	Natural Mapping Editor	Server Mapping Reqired	More Information
`REDEFINE`		yes	*Extracting Natural `REDEFINES`* under *Natural to IDL Mapping* in the IDL Extractor for Natural documentation
all	Rename of program	yes	*Renaming a Program*
all	Multiple interfaces	yes	*Extracting Multiple Interfaces*
all	Set constant	yes	*Setting Natural Parameters to Constants*
all	Suppress	yes	*Suppressing Natural Parameters*
Other combinations		no	

Natural Wrapper

Natural Wrapper	Server Mapping Required	More Information
IDL program name is not a valid Natural name and is therefore adapted, or the Natural program name is customized	yes	*Step 2: Customize Natural Server Names* under *Using the Natural Wrapper for the Server Side within NaturalONE*

Source Control of Server Mapping Files

Because server mapping files (Designer files with extension .cvm) contain text data only, a server mapping file is text-based (although it is not intended for human consumption). Therefore, you can include it in your source control management together with the IDL file and the Natural source(s) as a triplet that should always be kept in sync.

Comparing Server Mapping Files

For server mapping files (Designer files with extension .cvm), you can use a third party file/text compare tool to check if two files are identical.

A server mapping entry (corresponding to a line in a server mapping file) contains a creation timestamp at offset 276 (decimal) in the format *YYYYMMDDHHIISST*. The precision is 1/10 of a second.

5 Server Mapping Files for COBOL

- Server Mapping Files in the Designer 22
- How to Set the Type of Server Mapping Files 25
- When is a Server Mapping File Required? 26
- Migrating Server Mapping Files 31
- Source Control of Server Mapping Files 32
- Comparing Server Mapping Files 32

Server mapping enables the RPC server to correctly support special COBOL syntax such as `REDEFINE`s, `SIGN LEADING` and `OCCURS DEPENDING ON` clauses, `LEVEL-88` fields, etc. If one of these elements is used, the IDL Extractor for COBOL automatically extracts a server mapping file in addition to the IDL file (interface definition language). Also, the COBOL Wrapper may generate a server mapping file for RPC server generation. The server mapping is used at runtime to marshal and unmarshal the RPC data stream. There are client-side mapping files (Designer files with extension .cvm) and server-side mapping files (Designer files with extension .svm). If you have not used server-side mapping, we recommend you use client-side mapping.

See also *Server-side Mapping Files* in the respective RPC server documentation.

Server Mapping Files in the Designer

This section covers the following topics:

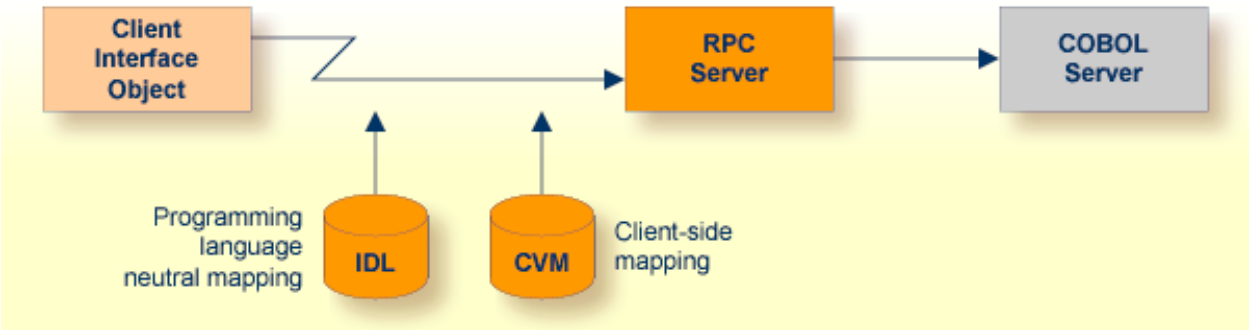
- Introduction
- Client-side Mapping
- Server-side Mapping

Introduction

The following rules apply to a server mapping file for COBOL in the Designer:

- The mapping file is of type client-side or server-side. We recommend using client-side mapping files, which is the default. You cannot use both types in the same Eclipse workspace. See ***How to Set the Type of Server Mapping Files***.
- Mapping files have the extension .cvm (client-side) or .svm (server-side).
- The mapping file has to relate to an appropriate IDL file. Always keep the IDL file and the mapping file together in the same folder.
- The mapping file is created if required by the COBOL Wrapper | Extractor. See ***When is a Server Mapping File Required?***.
- If an IDL file has a corresponding mapping file, at least one of the IDL programs in the IDL file requires server-mapping information to correctly call the target server. For those IDL programs, there is a server mapping (corresponding to a line) in the server mapping file.
- If there is an IDL file but no corresponding mapping file, there is no IDL program that requires server mapping information.

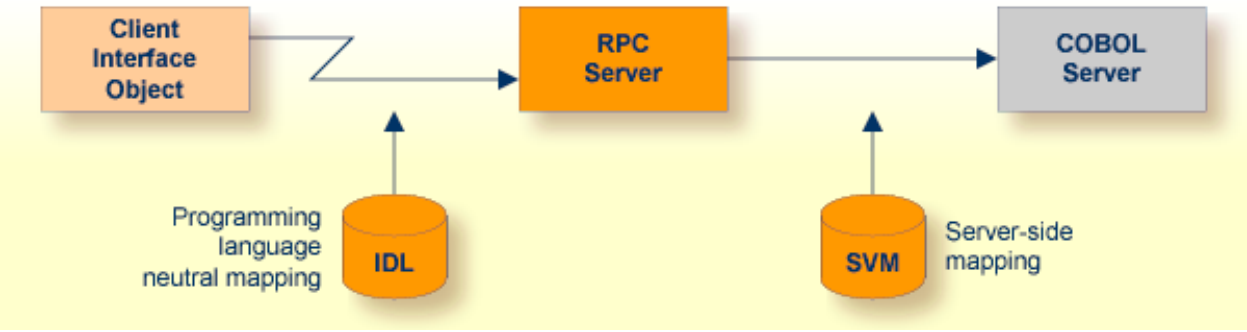
Client-side Mapping



The following rules apply to client-side mapping:

- Client-side mapping files have the extension .cvm in the Designer.
- The mapping files are generated when the Designer is in client-side mapping mode, see *How to Set the Type of Server Mapping Files*.
- Handling of client-side mapping is easier than server-side mapping. This can be an important criterion, for example if the RPC server is hosted in a mainframe environment and you do not have access to mainframe development resources. The following tasks are not required:
  - deploying the server mapping files to the RPC server
  - setting up a server-side mapping container in the mainframe environment
  - change management of server-side mapping files in the mainframe environment
- The mapping is wrapped into the client interface objects when an RPC client is generated. The client-side mapping must available before you creating any RPC client component, that is, the COBOL server program must be extracted or generated first.
- The mapping is sent during runtime to the target RPC server with the RPC request.
- Only point-to-point connections are allowed. The RPC client with the client-side mapping in its client interface object can only call one specific target COBOL server program. If you want to replace the COBOL server program by another implementation, for example Java, you need to rebuild all the RPC clients without the client-side mapping wrapped into their client interface object.
- Client-side mapping files are not supported by RPC clients generated with the DCOM Wrapper and COBOL Wrapper. Use server-side mapping files instead. See *Server-side Mapping*.
- To use client-side mapping, all EntireX components involved must be version 9.7 or higher.

Server-side Mapping



The following rules apply to server-side mapping:

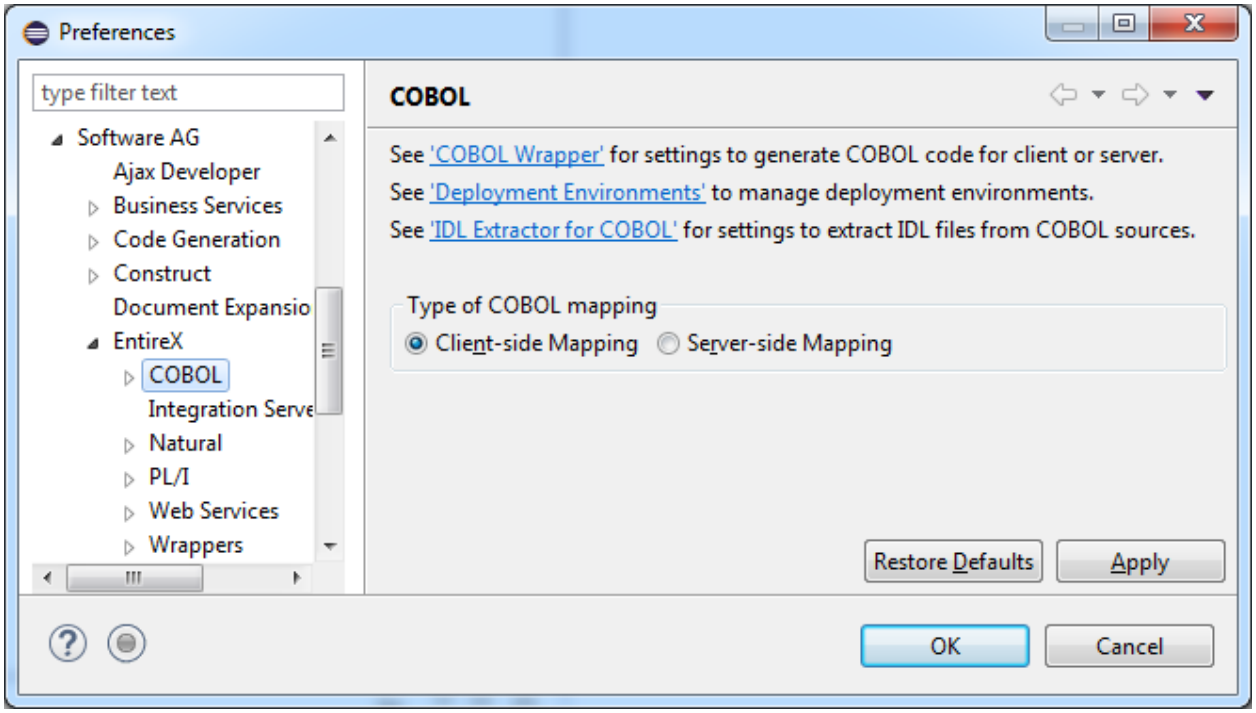
- Server-side mapping files have the extension .svm in the Designer.
- The files are generated when the Designer is in server-side mapping mode, see *How to Set the Type of Server Mapping Files*.
- The files have to be deployed to the RPC server. See *Deploying Server-side Mapping Files* in the respective RPC server documentation.

  Note: For IMS Connect and CICS ECI connections with the webMethods EntireX Adapter for Integration Server, server-side mapping files are not deployed. They are wrapped into the Integration Server connection - the same as client-side mapping files. For RPC connections, the server-side mapping files must be deployed to the target RPC server. See the *EntireX and your webMethods Integration Server Applications*.

- Server-side mapping is read at runtime by the RPC server from the server-side mapping container, see *Server-side Mapping Files* in the respective RPC server documentation
- Flexible connections are possible: the RPC client can call any target server program as long as the IDL interface matches. You do not need to rebuild the RPC clients when replacing the COBOL server program by a different implementation, for example Java, as required with client-side mapping files.
- Server-side mapping files can be migrated to client-side mapping files, see *Migrating Server Mapping Files*.

How to Set the Type of Server Mapping Files

Every Designer (Eclipse) workspace is either in client-side mapping mode (generating Designer server mapping files with extension .cvm) or server-side mapping mode (generating Designer server mapping files with extension .svm) for COBOL mapping. The default mode is client-side mapping and can be changed on the preferences page **Software AG > EntireX > COBOL**:



For more information on handling the different types of COBOL mapping, see *Server Mapping Files in the Designer*.

In *client-side* mapping mode, it is not possible to modify server-side mapping files (Designer files with extension .svm). This means that if there is a related server-side mapping file available for an IDL file, you cannot use the

- IDL Extractor for COBOL to extract an additional interface or modify an existing one
- COBOL Wrapper to generate an RPC Server

You have to migrate the server-side mapping files first. See *Migrating Server Mapping Files*.

In *server-side* mapping mode, it is not possible to modify client-side mapping files (Designer files with extension .cvm). This means that if there is a related client-side mapping file available for an IDL file, you cannot use the

- IDL Extractor for COBOL to extract an additional interface or modify an existing one

- COBOL Wrapper to generate an RPC Server

Later on, you can readjust the mode in the **Preferences** of the respective tool. See *COBOL Preferences* or *Generation Settings - Preferences* in the COBOL Wrapper documentation.

When is a Server Mapping File Required?

- IDL Extractor for COBOL
- COBOL Wrapper

IDL Extractor for COBOL

A server mapping file (Designer file with extension .cvm or .svm) is generated by the IDL Extractor for COBOL if the COBOL server program is of a specific interface type, contains specific COBOL syntax, or the IDL interface is redesigned (Suppress, Set constant, etc.) in the *COBOL Mapping Editor*. The type of server mapping (server-side with extension .svm or client-side with extension .cvm) is defined in a preference that applies to the entire Designer (Eclipse) workspace. See ***How to Set the Type of Server Mapping Files***.

Interface Type	COBOL Syntax	COBOL Mapping Editor	Server Mapping Required	More Information
CICS DFHCOMMAREA and In different to Out	all		yes	*CICS with `DFHCOMMAREA` Calling Convention - In different to Out* under *COBOL Mapping Editor in the IDL Extractor for COBOL documentation* (1) and the following COBOL server examples for CICS input message *different to* the CICS output message: ■ *Example 1: Redefines* ■ *Example 2: Buffer Technique* ■ *Example 3: COBOL `SET ADDRESS` Statements*
CICS Channel Container	all		yes	*CICS with Channel Container Calling Convention*
CICS Large Buffer	all		yes	*CICS with `DFHCOMMAREA` Large Buffer Interface*
COBOL Converter and In different to Out	all		yes	*COBOL Converter* (2)
IMS MPP (IMS Connect)	all		yes	*IMS MPP Message Interface (IMS Connect)*

Interface Type	COBOL Syntax	COBOL Mapping Editor	Server Mapping Required	More Information
IMS BMP	all		yes	*IMS BMP with Standard Linkage Calling Convention*
Micro Focus	`BINARY` clause, see *COBOL Data Type to Software AG IDL Mapping* under *COBOL to IDL Mapping* in the IDL Extractor for COBOL documentation		yes	*Micro Focus with Standard Linkage Calling Convention*
CICS DFHCOMMAREA, CICS Channel Container, CICS Large Buffer, IMS MPP (IMS Connect), COBOL Converter	`OCCURS` clause, see *COBOL Tables with Fixed Size*	Set Arrays (Fixed <-> Unbounded)	yes	See *Set Arrays (Fixed <-> Unbounded)* for interface type DFHCOMMAREA (In same as Out, In different to Out) | Large Buffer (In same as Out, In different to Out) | Channel Container | IMS Connect | COBOL Converter (In same as Out, In different to Out) [1, 2]
all	`OCCURS` clause, see *COBOL Tables with Fixed Size*	Map to	no	See *Map to* for interface type DFHCOMMAREA (In different to Out) | Large Buffer (In different to Out) | Channel Container | IMS Connect | COBOL Converter (In different to Out) [1,2]
		Map to In, Out, InOut		See *Map to In, Out, InOut* for interface type DFHCOMMAREA (In same as Out) | Large Buffer (In same as Out) | Batch | IMS BMP | Micro Focus | COBOL Converter (In same as Out) [1,2]
CICS DFHCOMMAREA, CICS Channel Container, CICS Large Buffer, IMS MPP (IMS Connect), COBOL Converter	COBOL group data items used in optional manner, see *Optional COBOL Group Data Items*	Set multiple possible output structures (MPO)	yes	See *Set Multiple Possible Output (MPO) Structures* for interface type DFHCOMMAREA (In same as Out, In different to Out) | Large Buffer (In same as Out, In different to Out) | Channel Container | IMS Connect | COBOL Converter (In same as Out, In different to Out) [1,2]
all	COBOL group data items	Map to	no	See *Map to* for interface type DFHCOMMAREA (In different to Out) | Large Buffer (In different to Out) | Channel Container | IMS Connect | COBOL Converter (In different to Out) [1,2]
		Map to In, Out, InOut		See *Map to In, Out, InOut* for interface type DFHCOMMAREA (In same as Out) | Large

Interface Type	COBOL Syntax	COBOL Mapping Editor	Server Mapping Required	More Information
				Buffer (In same as Out) | Batch | IMS BMP | Micro Focus | COBOL Converter (In same as Out) [1,2]
all	OCCURS DEPENDING ON clause, see *COBOL Tables with Variable Size - DEPENDING ON Clause*	Map to	yes	See *Map OCCURS DEPENDING ON* for interface type DFHCOMMAREA (In different to Out) | Large Buffer (In different to Out) | Channel Container | IMS Connect | COBOL Converter (In different to Out) [1,2]
		Map to In, Out, InOut		See *Map OCCURS DEPENDING ON* for interface type DFHCOMMAREA (In same as Out) | Large Buffer (In same as Out) | Batch | IMS BMP | Micro Focus | COBOL Converter (In same as Out) [1,2]
CICS DFHCOMMAREA, CICS Channel Container, CICS Large Buffer, IMS MPP (IMS Connect), COBOL Converter	REDEFINES clause, see *REDEFINES Clause*	Set multiple possible output (MPO) structures	yes	*Set Multiple Possible Output (MPO) Structures* for interface type DFHCOMMAREA (In same as Out, In different to Out) | Large Buffer (In same as Out, In different to Out) | Channel Container | IMS Connect [1,2]
		Map to		COBOL REDEFINES can be used in several ways: ■ *Select REDEFINE Paths* for interface type DFHCOMMAREA (In different to Out) | Large Buffer (In different to Out) | Channel Container | IMS Connect | COBOL Converter (In different to Out) ■ *Map to Multiple IDL Interfaces* for interface type DFHCOMMAREA (In different to Out) | Large Buffer (In different to Out) | Channel Container | IMS Connect | COBOL Converter (In different to Out)
		Map to In, Out, InOut		COBOL REDEFINES can be used in several ways: ■ *Select REDEFINE Paths* for interface type DFHCOMMAREA (In same as Out) | Large Buffer (In same as Out) | COBOL Converter (In same as Out ■ *Map to Multiple IDL Interfaces* for interface type DFHCOMMAREA (In same as Out) | Large Buffer (In same as Out) | COBOL Converter (In same as Out

Interface Type	COBOL Syntax	COBOL Mapping Editor	Server Mapping Required	More Information
Batch, IMS BMP, Micro Focus	`REDEFINES` clause, see *`REDEFINES` Clause*	Map to In, Out, InOut	yes	COBOL `REDEFINES` can be used in serveral ways: ■ *Select REDEFINE Paths* for interface type Batch | IMS BMP | Micro Focus ■ *Map to Multiple IDL Interfaces* for interface type Batch | IMS BMP | Micro Focus
all	`SIGN LEADING [SEPARATE]` clause, see *`SIGN LEADING` and `TRAILING SEPARATE` Clauses*		yes	
all	`SIGN TRAILING [SEPARATE]` clause, see *`SIGN LEADING` and `TRAILING SEPARATE` Clauses*		yes	
all	`SYNCHRONIZED` clause, see *`SYNCHRONIZED` Clause*		yes	
all	all	Rename of program	yes	See *Map to Multiple IDL Interfaces* for interface type DFHCOMMAREA (In same as Out, In different to Out) | Large Buffer (In same as Out, In different to Out) | Channel Container | Batch | IMS BMP | Micro Focus | IMS Connect | COBOL Converter (In same as Out, In different to Out) (1,2)
all	■ *Condition Names - Level-88 Data Items* ■ *COBOL Data Items Expecting Single Constant Values*	Set to constant	yes	See *Set COBOL Data Items to Constants* for interface type DFHCOMMAREA (In same as Out, In different to Out) | Large Buffer (In same as Out, In different to Out) | Channel Container | Batch | IMS BMP | Micro Focus | IMS Connect | COBOL Converter (In same as Out, In different to Out) (1,2)
all	See *Unneeded COBOL Data Items*	Suppress	yes	*Suppress Unneeded COBOL Data Items* for interface type DFHCOMMAREA (In same as Out, In different to Out) | Large Buffer (In same as Out, In different to Out) |

Interface Type	COBOL Syntax	COBOL Mapping Editor	Server Mapping Required	More Information
				Channel Container | Batch | IMS BMP | Micro Focus | IMS Connect | COBOL Converter (In same as Out, In different to Out) (1,2)
other combinations			no	

Notes:

1. For interface types DFHCOMMAREA and Large Buffer, COBOL server programs use either the same data structure on input and output ("In same as Out"), or overlay the input data structure with a different output data structure ("In different to Out"). See *COBOL Mapping Editor*.
2. For interface type COBOL Converter, COBOL input and output is either described by the same layout ("In same as Out") or the input is overlaid by a different output layout ("In different to Out"). See *COBOL Mapping Editor*.

COBOL Wrapper

A server mapping file (Designer file with extension .cvm or .svm) is generated by the COBOL Wrapper if an RPC server is generated (1,2) and at least one IDL program meets the criteria in the table below. The type of server mapping (server-side with extension .svm or client-side with extension .cvm) is defined in a preference that applies to the entire Designer (Eclipse) workspace. See ***How to Set the Type of Server Mapping Files***.

Interface Type	IDL Type	COBOL Wrapper	Server Mapping Required	More Information
CICS Large Buffer	all		yes	*CICS with `DFHCOMMAREA` Large Buffer Interface* under *COBOL Server Interface Types* in the COBOL Wrapper documentation
CICS Channel Container	all		yes	*CICS with Channel Container Calling Convention*
IMS BMP	all		yes	*IMS BMP with Standard Linkage Calling Convention*
Micro Focus	I2 or I4		yes	*Micro Focus with Standard Linkage Calling Convention* | *IDL Data Types*
all	IDL unbounded array		yes	`array-definition` under *Software AG IDL Grammar* in the IDL Editor documentation

Interface Type	IDL Type	COBOL Wrapper	Server Mapping Required	More Information
all	IDL unbounded group		yes	`group-parameter-definition` under *Software AG IDL Grammar* in the IDL Editor documentation
all	all	IDL program name is not a valid COBOL name and is therefore adapted, or the COBOL program name is customized	yes	*Customize Automatically Generated Server Names* under *Generating COBOL Source Files from Software AG IDL Files* in the COBOL Wrapper documentation
other combinations			no	

(1) Server mapping files are never generated for RPC clients.

Migrating Server Mapping Files

This section covers the following topics:

- Introduction
- Step 1: Rename the Server-side Mapping File
- Step 2: Remove the Server-side Mapping Files on Target RPC Server
- Step 3: Rebuild and Deploy all RPC Clients

Introduction

Server-side mapping files (Designer files with extension .svm) can be migrated to client-side mapping files (Designer files with extension .cvm).

Note: Client-side mapping files are not supported by RPC clients generated with the DCOM Wrapper and COBOL Wrapper.

All EntireX components involved in the migration must be version 9.7 or higher:

- the target RPC server z/OS (CICS | Batch | IMS), CICS ECI, IMS Connect, Micro Focus)
- the webMethods EntireX Adapter for Integration Server. See *EntireX and your webMethods Integration Server Applications*
- the RPC client runtimes

Step 1: Rename the Server-side Mapping File

Rename the extension .svm to .cvm in the Designer. This results in a client-side mapping file.

Step 2: Remove the Server-side Mapping Files on Target RPC Server

Remove the server-side mapping files in the server-side mapping container of the target RPC server. See *Undeploying Server-side Mapping Files* in the respective RPC server documentation.

Note: For IMS Connect and CICS ECI connections with webMethods EntireX Adapter for Integration Server, this step is not required.

Step 3: Rebuild and Deploy all RPC Clients

1. Re-create (wrap, build, compile etc.) all involved RPC clients [(1,2)] using the related IDL file to wrap the client-side mapping files into the client interface objects. See ***EntireX Wrappers***.
2. Test the RPC clients with client-side mapping files.
3. If necessary, re-deploy the RPC clients with client-side mapping files.

[(1)] This includes all variants of RPC connections of the webMethods EntireX Adapter for Integration Server. See *webMethods Integration Server Wrapper*.

Source Control of Server Mapping Files

Because server mapping files (Designer files with extension .cvm or .svm) contain text data only, a server mapping file is text-based (although it is not intended for human consumption). Therefore, you can include it in your source control management together with the IDL file and the COBOL source(s) as a triplet that should always be kept in sync.

Comparing Server Mapping Files

For server mapping files (Designer files with extension .cvm or .svm), you can use a third party file/text compare tool to check if two files are identical.

A server mapping entry (corresponding to a line in a server mapping file) contains a creation timestamp at offset 276 (decimal) in the format `YYYYMMDDHHIISST`. The precision is 1/10 of a second.

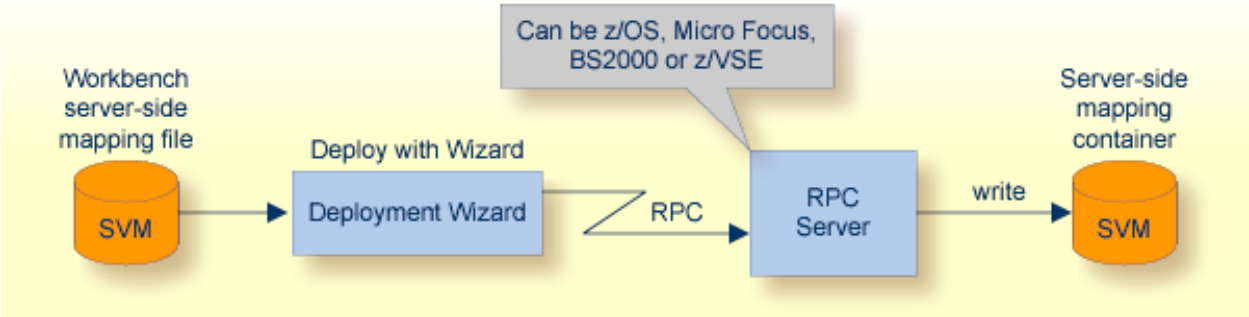
6 Server Mapping Deployment Wizard

- Introduction 35
- Using 35
- Preferences 39
- Command-line Mode 42

See also *Deploying Server-side Mapping Files* for CICS ECI | IMS Connect.

Note: To deploy server mapping files to the webMethods EntireX Adapter for Integration Server you need to update your Adapter connection. See *Step 3: Create or Update an Adapter Connection* in the Integration Server Wrapper documentation.

Introduction



Deploying with the wizard requires an active RPC server. Also, the Deployment Service of the RPC server must be properly configured. See the platform-specific documentation for more information:

- z/OS, see *Deployment Service* for CICS | Batch | IMS in the respective RPC server documentation.
- UNIX or Windows for Micro Focus COBOL, see *Deployment Service* in the RPC Server for Micro Focus documentation.
- BS2000, see *Deployment Service* in the RPC Server for BS2000 documentation.
- z/VSE, see *Deployment Service* for CICS | Batch in the respective RPC server documentation.

Using

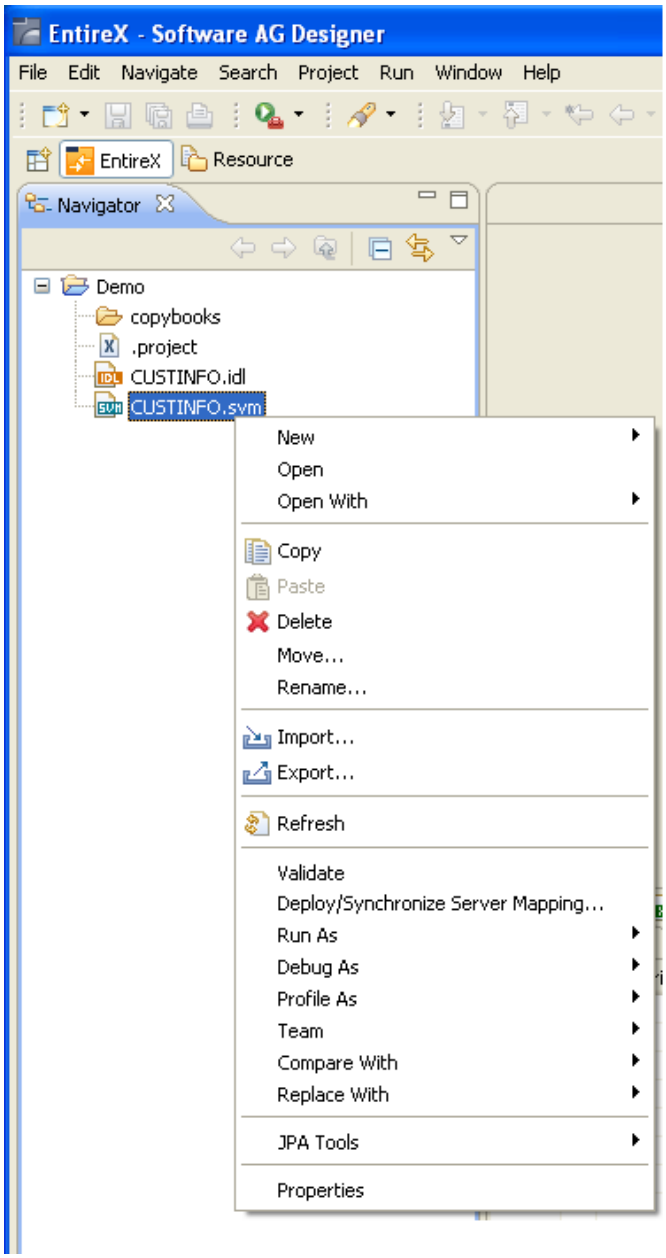
To deploy a server-side mapping file with the wizard, follow the steps below:

- Step 1: Start the Wizard
- Step 2a: Create a New Deployment Environment
- Step 2b: Define the Connection to the Deployment Service and Deploy
- Step 3: Select and Existing Deployment Environment and Deploy

A server-side mapping file is a Designer file with extension .svm. See ***Server Mapping Files for COBOL***.

Step 1: Start the Wizard

To start the Server Mapping Deployment Wizard, select a server-side mapping file and from the context menu choose **Deploy/Synchronize Server Mapping...**. Alternatively, you can use the context menu of the IDL file and choose **COBOL > Deploy/Synchronize Server Mapping...**.



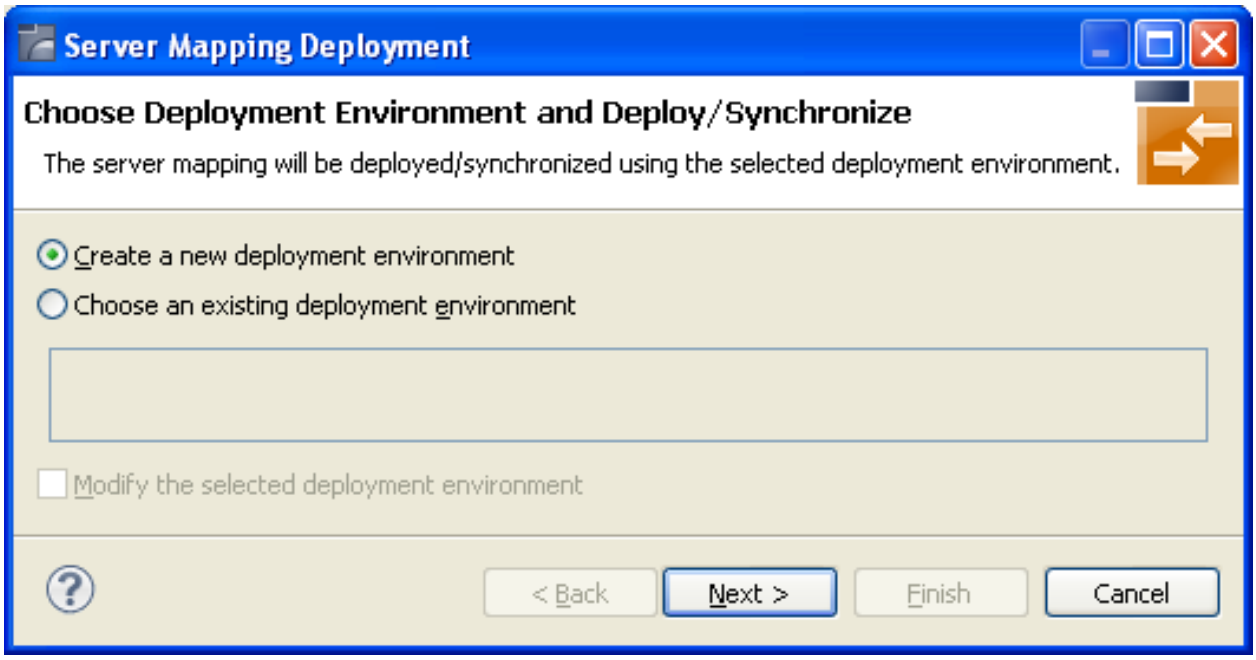
To continue, press **Next** with one of the following choices:

- If you are using the Server Mapping Deployment Wizard for first time with no predefined deployment environment preferences, continue with *Step 2a: Create a New Deployment Environment* below.

- If deployment environments are already defined, you may also continue with *Step 3: Select and Existing Deployment Environment and Deploy*.

Step 2a: Create a New Deployment Environment

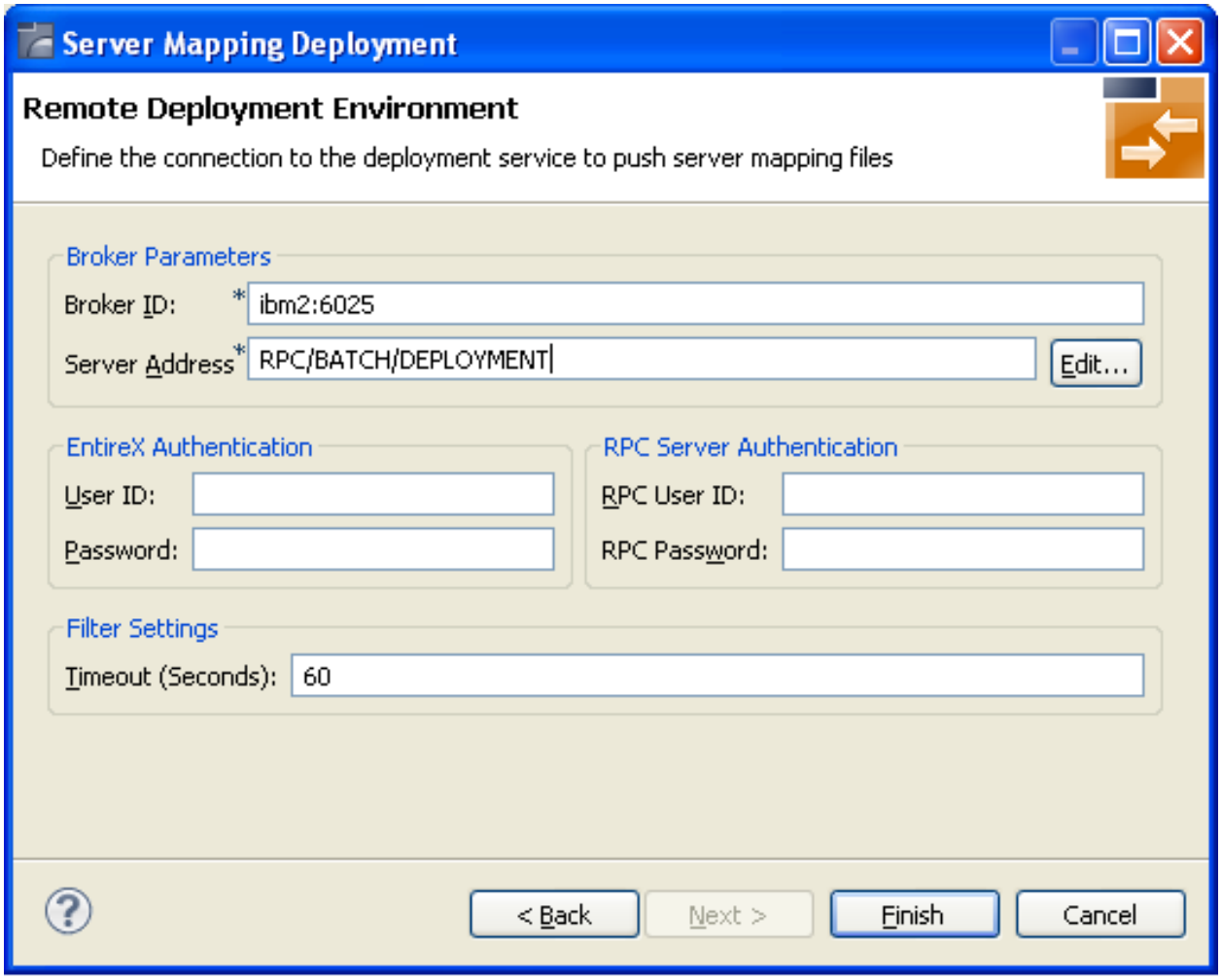
If no deployment environments are defined, you only have the option to create a new deployment environment.



Select **Create a new deployment environment** and press **Next** to continue with *Step 2b: Define the Connection to the Deployment Service and Deploy*.

Step 2b: Define the Connection to the Deployment Service and Deploy

Use this page to define a connection to the deployment service of the RPC server.



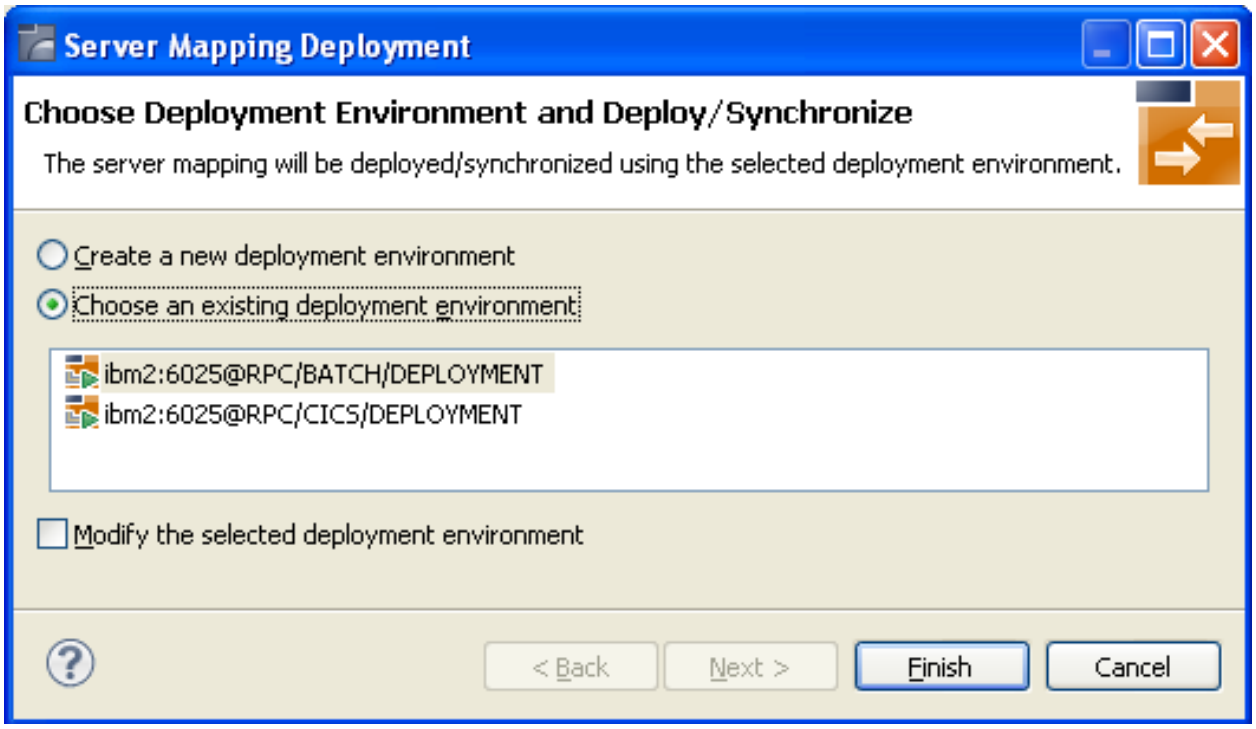
Enter the required fields:

1. **Broker Parameters** Broker ID and Server Address, which will have the default format. The last part (broker service) of the server address must always be "DEPLOYMENT".
2. The **EntireX Authentication** parameters describe the settings for the broker. These parameters apply if the *broker* is running with *EntireX Security*.
3. The **RPC Server Authentication** parameters describe the settings for the RPC server. These parameters apply if the *RPC server* is running with security. See Impersonation in the respective RPC Server documentation.
4. The given **Timeout** value must be in the range from 1 to 9999 seconds (default: 60).

Press **Finish** to deploy. Deployment of the server mapping is successful if the wizard ends. No confirmation message is given.

Step 3: Select and Existing Deployment Environment and Deploy

Use this page to select the deployment environment (i.e. the RPC server) to which you want to deploy.



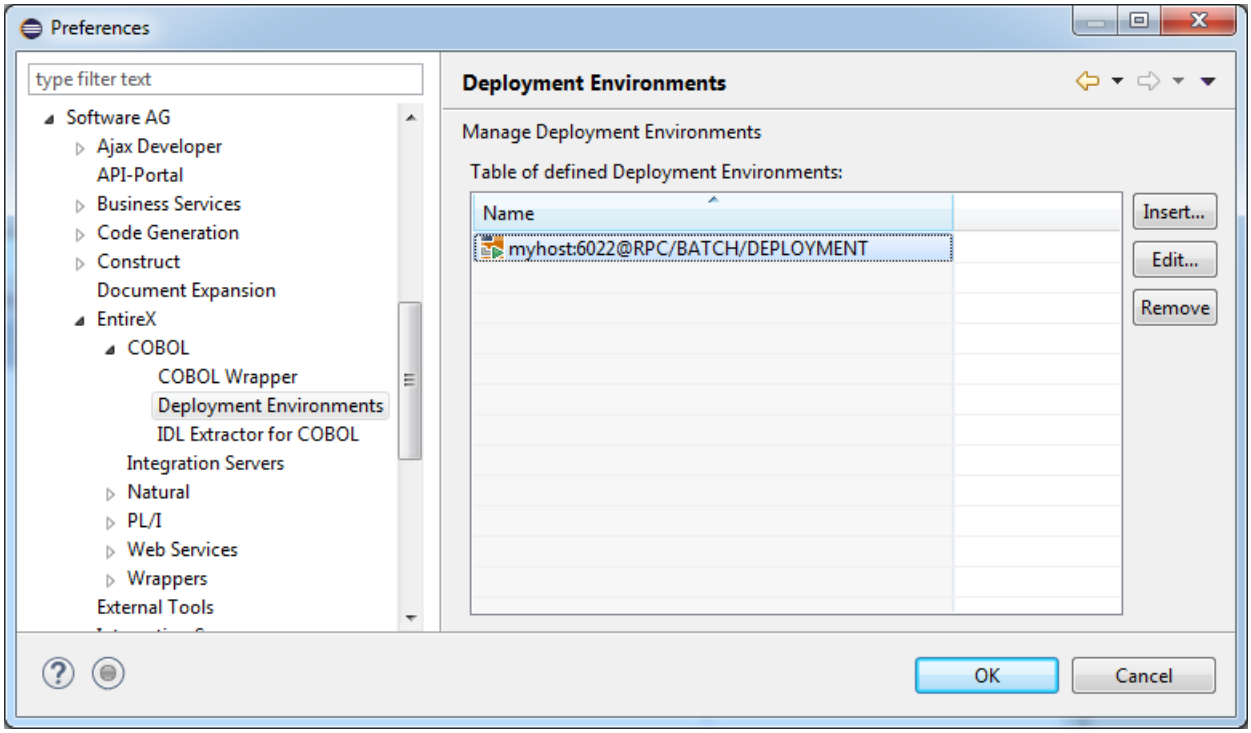
Check the option **Choose an existing deployment environment** and select a deployment environment from the list. Press **Finish** to deploy. Deployment is successful if the wizard ends. No confirmation message is given.

Preferences

In the preferences for the Server Mapping Deployment Wizard you define deployment environments, a connection to the Deployment Service of the RPC server. See *Deployment Service* in the respective RPC server sections of the documentation The following sections are offered:

- Create a New Deployment Environment
- Edit an Existing Deployment Environment
- Remove an Existing Deployment Environment

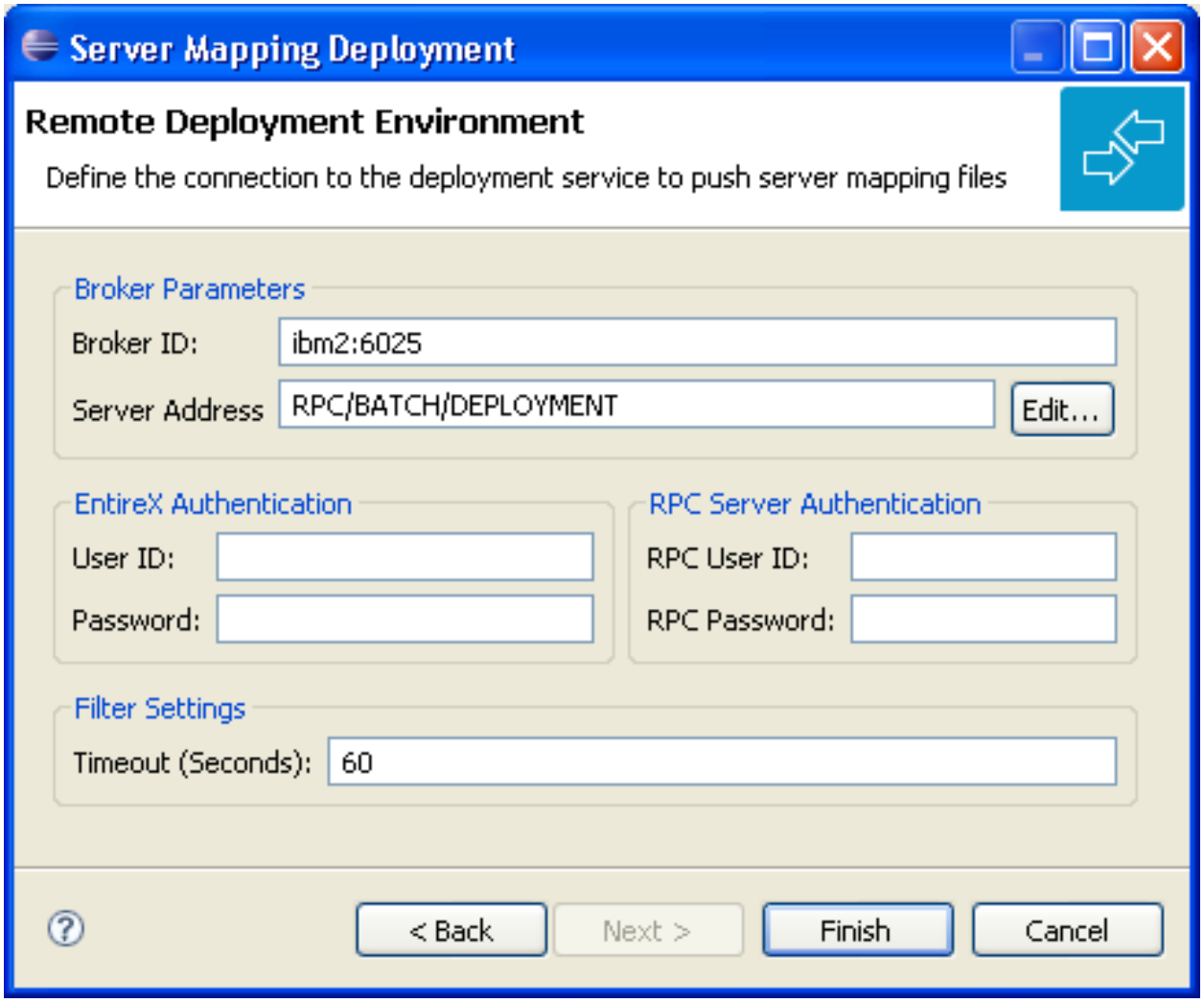
The deployment environment is managed from the deployment environment **Preferences** page. The deployment environments can be created, edited and removed. The deployment environment will be used for the selection lists in the Deployment Wizard. To manage these deployment environments, open the **Preferences** page:



Create a New Deployment Environment

To create a new deployment environment

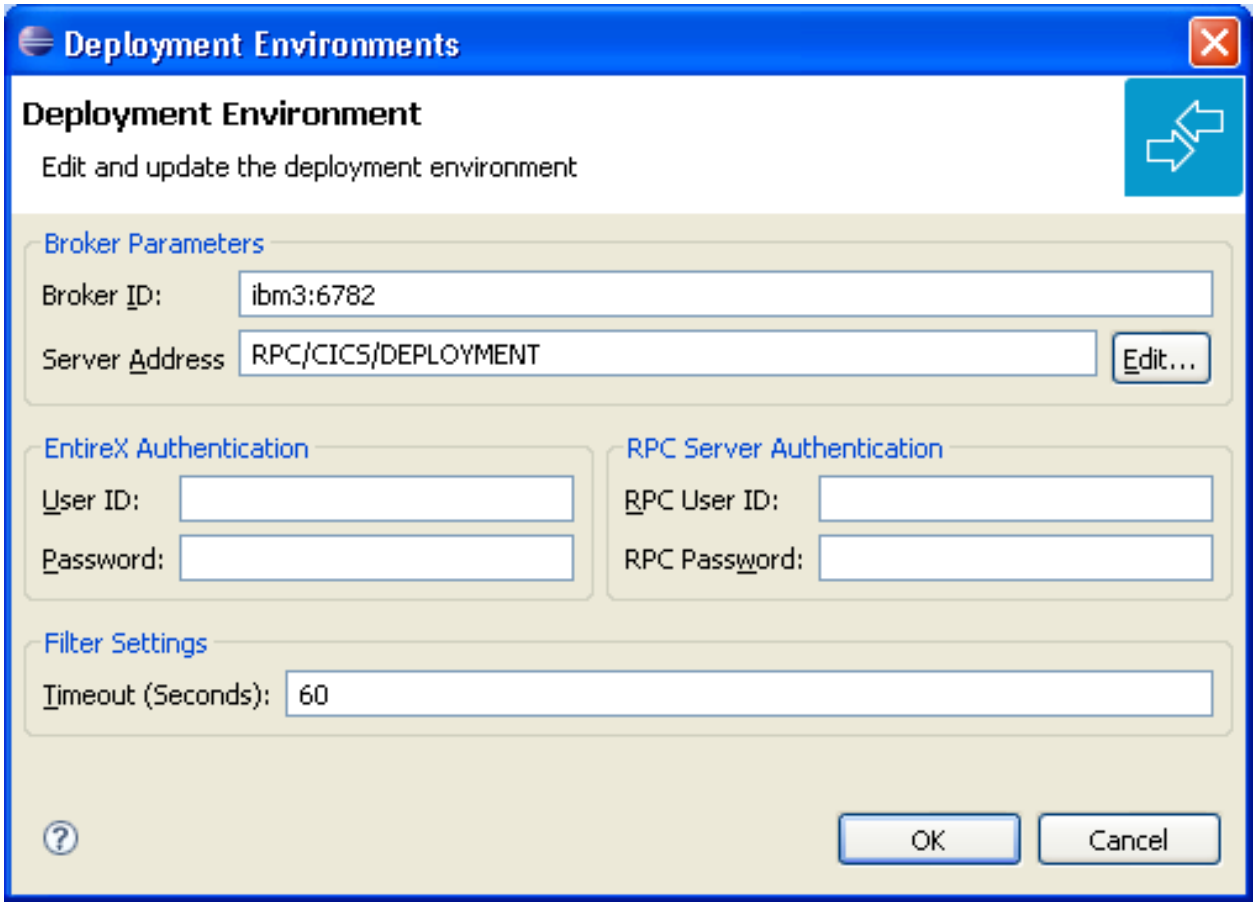
- Press **Insert**.



Edit an Existing Deployment Environment

To edit an existing deployment environment

- Select the table row and press **Edit**. If multiple entries are selected, the first entry is used.



Remove an Existing Deployment Environment

To remove an existing deployment environment

- Select the table row and press **Remove**. Multiple selections are possible.

Command-line Mode

The command `-deploy:cobol` is provided to deploy mapping information contained in server-side mapping files, using the Designer in command-line mode. See *Using EntireX in the Designer Command-line Mode* for general information.

To undeploy previously deployed server mapping information, remove the server-side mapping file (.svm) and execute the command `-deploy:cobol` with the IDL file only.

Command-line Options

Task	Command	Option	Description
Deploy SVM files.	`-deploy:cobol`	`-environment`	Target environment. Name of the COBOL deployment environment or an RPC server description.
		`-brokeruser`	User used for broker authentication (optional).
		`-brokerpassword`	Password used for broker authentication (optional).
		`-rpcuser`	User used for RPC server authentication (optional).
		`-rpcpassword`	Password used for RPC server authentication (optional).

Note: Run the command from the directory containing the IDL file and the corresponding server-side mapping file (.svm). If no server-side mapping file is found, the previously deployed server mapping information related to the IDL file will be removed on the server side (undeployed). See *Undeploying Server-side Mapping Files* in the respective sections of the documentation.

Example

```
-deploy:cobol /SVMDeployTests/idls/basicodo.idl  /SVMDeployTests/idls/basicdt.idl ↩
/SVMDeployTests/idls/basicarr.idl
 -environment ibm2:3980@RPC/RPCALL/DEPLOYMENT
 -brokeruser EXXUSR1
 -brokerpassword EXX$PWD1
```

7 EntireX IDL Tester

- Calling the IDL Tester 46
- Using the Broker and RPC User ID/Password 48

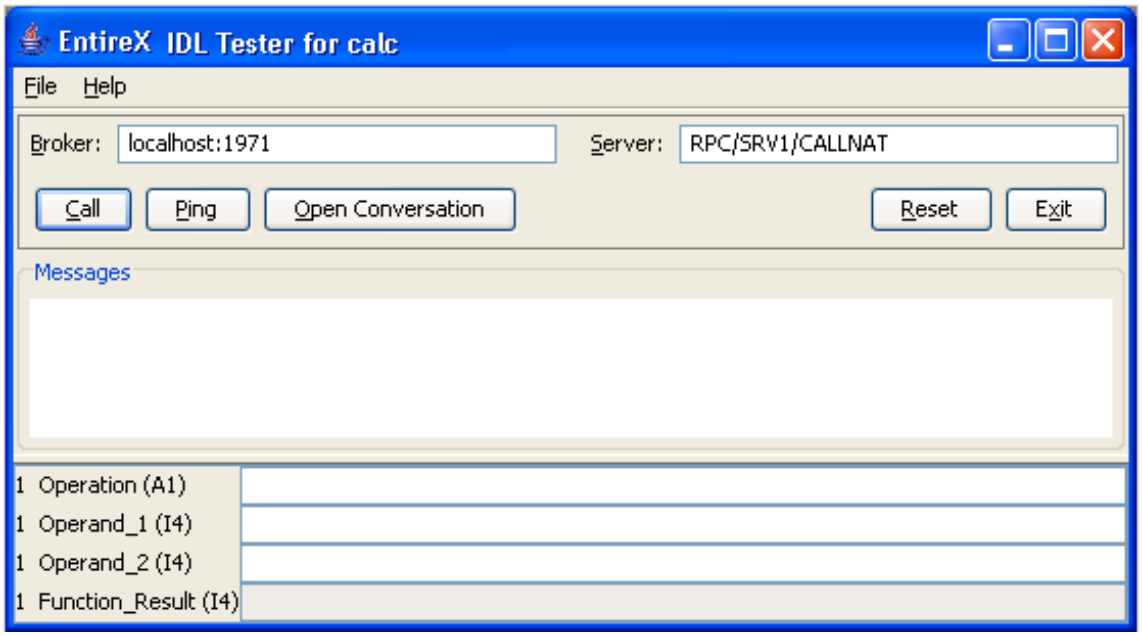
The IDL Tester is an easy-to-use utility to communicate with *EntireX RPC Servers*. It acts as an RPC client application, taking as input an IDL file. It is useful for testing extracted or generated artifacts such as the IDL file, XML mapping file for Web services, server mapping files Natural | COBOL etc. together with the RPC Server. It verifies the mapping of the data and all involved components up to the endpoint (RPC Server, Web Service, IBM MQ, etc.).

Note: With the Java Wrapper you can also generate IDL-related test classes into a Java project. See *Using the IDL Tester* under *Using the Java Wrapper*.

Calling the IDL Tester

To call the IDL Tester

- Select an IDL file, and from the context menu, choose **Software AG IDL Tester**. This launches the IDL Tester.



The dialog contains fields for Broker ID and server address and buttons, a message area for general output and error messages, and a parameter area with the input and output parameters. The fields for the input parameters have a white background and the fields for the output parameters are disabled and have a gray background.

Broker ID and server address are displayed as specified in the properties of the IDL file; they can be modified. To modify the server address, it is sufficient to enter the server; RPC/CALLNAT will be added automatically.

Button	Explanation
Call	Perform the remote procedure call.
Ping	Ping the broker and the RPC server.
Open Conversation	Open a conversational RPC (only visible if no conversation is open).
Commit Conversation	Close the conversational RPC and commit the conversation (only visible after **Open Conversation**).
Abort Conversation	Close the conversational RPC and abort the conversation (only visible after **Open Conversation**).
Reset	Reset the message area and the parameters.
Exit	Exit the IDL Tester.

The bottom of the dialog contains text fields for the parameters of the Software AG IDL file. These fields are prefixed with the group level, field name and type. `IN` and `INOUT` parameters can be edited, `OUT` parameters cannot be modified. The fields for `INOUT` and `OUT` parameters are updated after a successful call. Array elements must be separated by a semicolon. The last array element behind a semicolon will be used to fill up the array completely, e.g. (I4/5) and value 1;2 results in 1;2;2;2;2, but value 1;2; results in 1;2;;;.

The **File** Menu contains the entries **Options** and **Exit**. The **Options** dialog allows you to set user credentials. See *Using the Broker and RPC User ID/Password*.

To execute the remote call via EntireX Broker

- Use the **Call** button or press Return in one of the input parameter fields.

If the Broker returns errors (a BrokerException is thrown), the error message is displayed in the Messages area.

To ping the Broker and the RPC server

- Use the **Ping** button.

To delete the entries in the message area and the parameter fields

- Use the **Reset** button.

Using the Broker and RPC User ID/Password

EntireX supports two user ID/password pairs: a broker user ID/password pair and an (optional) RPC user ID/password pair sent from RPC clients to RPC servers. With EntireX Security, the broker user ID/password pair can be checked for authentication and authorization.

The RPC user ID/password pair is designed to be used by the receiving RPC server. This component's configuration determines whether the pair is considered or not. Useful scenarios are:

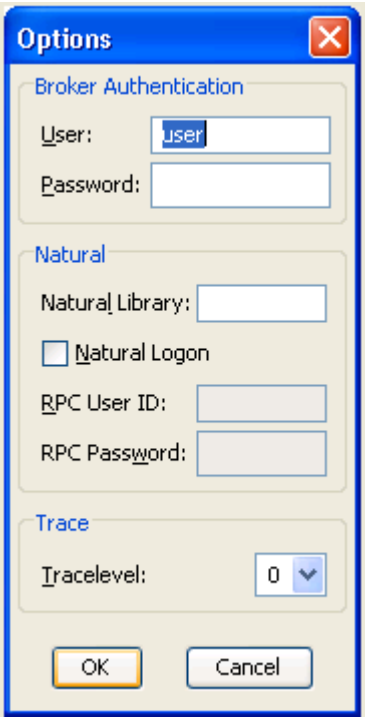
- Credentials for Natural Security
- Impersonation in the respective RPC Server documentation
- Web Service Transport Security with the RPC Server for XML/SOAP, see *XML Mapping Files*
- Service execution with client credentials for EntireX Adapter Listeners, see *Configuring Listeners*
- etc.

Sending the RPC user ID/password pair needs to be explicitly enabled by the RPC client. If it is enabled but no RPC user ID/password pair is provided, the broker user ID/password pair is inherited to the RPC user ID/password pair.

With the check box **Natural Logon**, sending the RPC user ID/password pair is enabled for the IDL Tester. If you do so, we strongly recommend using SSL/TLS. See *SSL/TLS and Certificates with EntireX*.

To use the broker and RPC user ID/password

1 Specify `User` and `Password` as **Broker Authentication** in the **Options** dialog.



2 Check the check box **Natural Logon** to enable sending the RPC user ID/password pair.

3 If different user IDs and/or passwords are used for broker and RPC, enter `RPC User ID` and `RPC Password` to provide a different RPC user ID/password pair.

4 By default the library name sent to the RPC server is retrieved from the IDL file (see `library-definition` under *Software AG IDL Grammar* in the IDL Editor documentation). The library name can be overwritten. This is useful if communicating with a Natural RPC server. Specify a `Natural Library`.

8 Using EntireX Custom Wrappers

- Define EntireX Custom Wrappers 52
- Running a Custom Wrapper 59

The EntireX Custom Wrappers are user-configurable, template-based wrappers and need:

- a Software AG IDL file
- a template (for example, client or server)
- the IDL Compiler

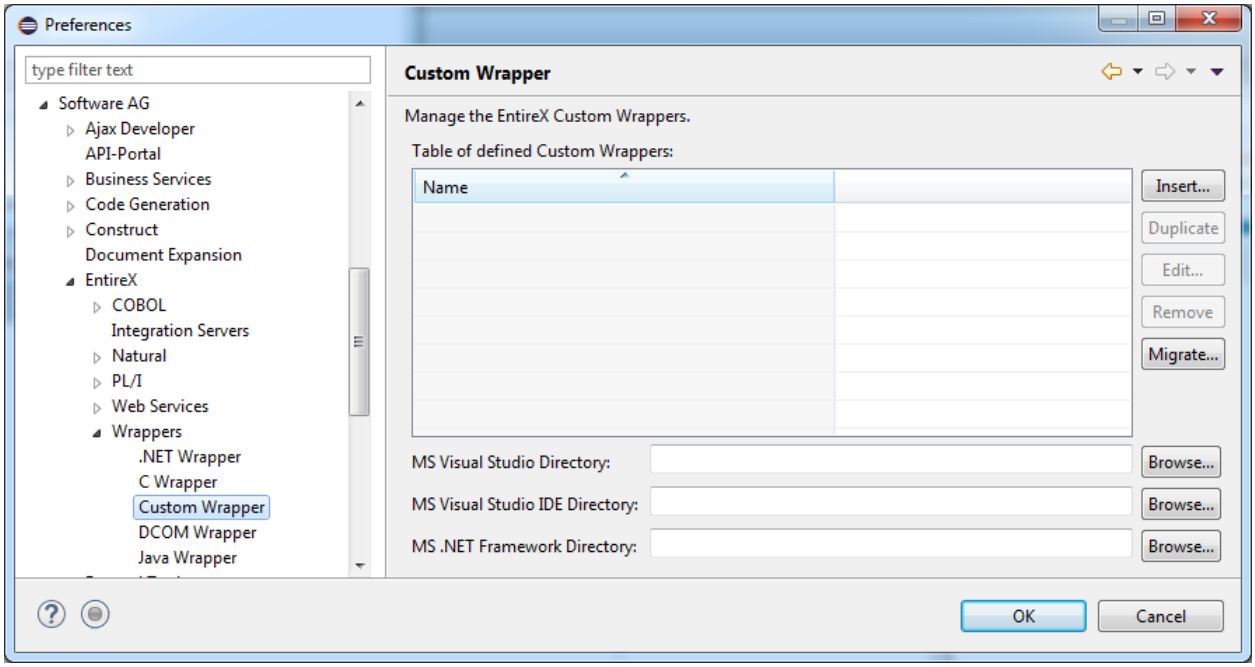
Naturally, existing *.plugin* files from former EntireX installations, called Designer Plug-ins, can be migrated. The Custom Wrapper definitions are stored in the current Eclipse workspace and can be managed in the preferences. Any changes in the Custom Wrapper definitions require a restart of the Designer to take effect. This chapter covers the following topics:

Define EntireX Custom Wrappers

The EntireX Custom Wrappers are managed in the Custom Wrapper preferences page.

The central panel lists all known (active or inactive) Custom Wrappers, that is, all Custom Wrappers that are found in the current workspace. You can create a new empty Custom Wrapper, copy, edit or remove an existing Custom Wrapper or migrate your existing *.plugin* file from a former EntireX installation.

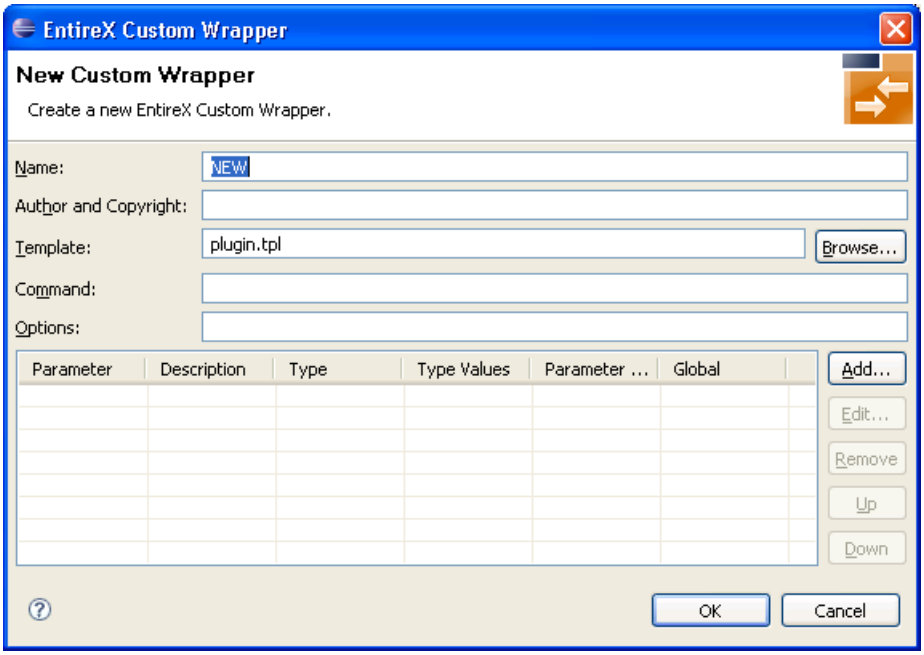
- Creating a New Custom Wrapper
- Parameters
- Parameter Values
- Wildcards



Creating a New Custom Wrapper

To create a new Custom Wrapper

1 Choose **Insert...**



2 Enter the fields. The Name is a required field and must be unique, because it is the identifier for the Custom Wrapper. This name occurs on the Custom Wrapper Properties page and will be used for the command line with the prefix minus sign.

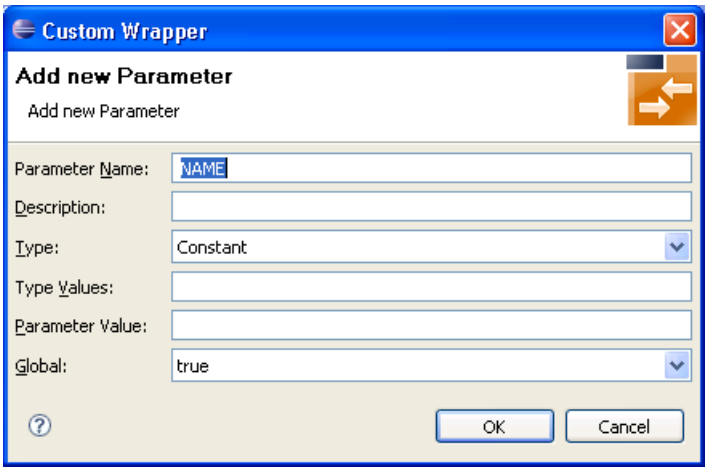
Field	Description
Name	Used as the Custom Wrapper menu item text.
Author and copyright	Important if you want to share your Custom Wrapper with other users.
Template	Name of the template file to be used by the *Software AG IDL Compiler*. Use a fully qualified file name.
Command	Executable command file name. Default is empty and means the *Software AG IDL Compiler* will be executed.
Options	Batch parameters. They are sent to the *Software AG IDL Compiler*.
Parameter	Parameters (constant or modifiable) to call the Custom Wrapper.

3 Optionally, add some parameters as described in the following sections.

To add parameters

- Choose **Add...**

  A new dialog with default values is displayed. See section *Parameters* for a description of the individual fields. Modifiable parameters automatically appear with an appropriate label and GUI widget on the Custom Wrapper IDL properties tab.



To edit parameters (see *Parameters*)

1. Select the parameter to edit in the table.
2. Choose **Edit...**

   or

   Double click on the parameter in the table.

   A new dialog with the current parameter values is displayed. See section *Parameters* for a description of the individual fields. For all strings in the fields **Type Values** and **Parameter Value**, you can also use wildcards. Wildcards are like variables for actual values that may change with the platform or in other properties tabs. See the list of *Wildcards*.
3. Choose **OK** to save your entries and close the dialog or **Cancel** to close the dialog without saving.

To remove parameters

1. Select the parameters to remove in the table.
2. Choose **Remove**.

   Multiple selections are possible.

To change the parameter order

1 Select the parameter to edit in the table.

2 Use the **Up** and **Down** buttons to change the order of the parameters.

Parameters

Parameters consist of the following fields:

Field	Purpose
Parameter	The parameter identifier, as required by the *erxidl* template (must not contain whitespace characters; often in uppercase).
Description	The textual description of this parameter, as displayed in the associated Custom Wrapper IDL properties tab. Human-readable, may contain whitespace characters and short examples.
Type	Dialog box of possible GUI representations of the ***Parameter Values***.
Type Values	The possible values that may be assigned to the parameter, in the form `<parametername>=<value>`. Values are separated by a semicolon.
Parameter Value	Default value for the GUI representation, must be one of the values listed in the **Type Values** field.
Global	Flag for the parameter value. True (default) means the parameter value is for all IDL files, false means the parameter value is IDL specific (stored in IDL properties).

Parameter Values

Parameter “types” are GUI representations of the parameter values in the IDL properties tab for this Custom Wrapper. They can be:

Entry	Purpose, Usage
Check box	Will draw a check box in the Custom Wrapper IDL **properties** tab. **Type Values** must have two values, separated by a semicolon. The first value is taken if the check box is checked, the second value is taken if the check box is cleared.
Combo box	Will draw a combo box (drop-down list) in the Custom Wrapper IDL properties tab. **Type Values** must have at least one entry, or multiple entries separated by semicolon. An entry may be: ■ a simple string (without the “=” character), which is displayed as an entry in the box and is sent to the Software AG IDL Compiler. ■ a pair "`<val>=<string>`", where `<string>` is displayed in the combo box, but `<val>` is sent to the Software AG IDL Compiler.
Constant	Will not generate a GUI element in the associated IDL **properties** tab. **Type Values** just has one string, which is sent to the Software AG IDL Compiler.

Entry	Purpose, Usage
Text	Will show a text field in the associated Custom Wrapper IDL properties tab. The text field content is taken for the Software AG IDL Compiler command-line construction. If whitespace characters are typed in the text field, the Software AG IDL Compiler command-line portion is protected by double quotes. Type values may contain a string representing as default value for the text field.

Wildcards

Wildcards start with the % (percent sign). Implemented wildcards:

Wildcard Name	Purpose	Example Value
`%brokerid`	Get the Broker ID from the **General** tab.	localhost:1971
`%server`	Get the server address from the **General** tab.	RPC/SRV1/CALLNAT
`%serverclass`	Get the server class identifier from the **General** tab.	RPC
`%servername`	Get the server name from the **General** tab.	SRV1
`%service`	Get the service identifier from the **General** tab.	CALLNAT
`%idlfile`	Get the current IDL file (as absolute path name) as selected in the Designer.	C:\Examples\example.idl
`%idlname`	Get the current IDL file (just the file name with extension, no path) as selected in the Designer.	example.idl
`%pureidlname`	Get the current IDL file name without path and without the file extension.	example
`%idlpath`	Get the current IDL file path (without the IDL file name) as selected in the Designer.	C:\Examples
`%xmlfile`	Get the current XML mapping file name from the XML tab.	C:\Examples\example.xmm
`%msdotnetenv`	Path of Microsoft Visual Studio environment as set in the **Tools Options** menu item.	C:\Program Files\Microsoft Visual Studio 10\Common7\IDE\
`%netfrmdir`	Path for installation of the .NET framework.	C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727
`%version`	Get the EntireX version.	8.0
`%osname`	Get the operating system name, taken from the Java system property *os.name*.	Windows XP
`%osarch`	Get the operating system architecture, taken from the Java system property *os.arch*.	x86
`%osversion`	Get the operating system version, taken from the Java system property *os.version*.	5.1
`%fileseparator`	Get the platform-specific file separator character, taken from the Java system property *file.separator*.	\

Wildcard Name	Purpose	Example Value
`%fileencoding`	Get the platform-specific file encoding, taken from the Java system property *file.encoding*.	Cp1252
`%pathseparator`	Get the platform-specific path separator character, taken from the Java system property *path.separator*.	;
`%username`	Get the current user name, taken from the Java system property *user.name*.	administrator
`%userhome`	Get the assigned user home directory, taken from the Java system property *user.home*.	C:\Documents and Settings\administrator
`%userdir`	Get the current user directory, taken from the Java system property *user.dir*.	C:\Documents and Settings\administrator\My Documents

To duplicate an existing Custom Wrapper

1 Select the Custom Wrapper in the list.

2 Proceed as described in *Creating a New Custom Wrapper* and press **Duplicate** instead of **Insert...**.

A new Custom Wrapper named *Copy of* <*name*> is displayed in the list.

3 Modify the entries as described in *Creating a New Custom Wrapper*.

To edit an existing Custom Wrapper

1 Select the Custom Wrapper in the list.

2 Proceed as described in *Creating a New Custom Wrapper* and press **Edit...** instead of **Insert...**.

3 Modify the entries as described in *Creating a New Custom Wrapper*.

To remove an existing Custom Wrapper

1 Select the Custom Wrapper in the list.

2 Press **Remove**.

To migrate an existing *.plugin* file from a previous EntireX Installation

1 Proceed as described in *Creating a New Custom Wrapper* and press **Migrate...** instead of **Insert...**.

2 Browse to the location of the *.plugin* file and select the file to migrate.

3 In the ensuing dialog, modify the entries as described in *Creating a New Custom Wrapper*.

4 Make sure the template file can be accessed correctly.

Running a Custom Wrapper

To start the Custom Wrapper in GUI

1 Select an IDL file.

2 Open the context menu, choose **Other > Generate Via Template** and select the desired item (for example, **New**).

When the Custom Wrapper is started, the Designer starts the Software AG IDL Compiler and feeds it with a template, the IDL file name and (optional) parameters.

The parameters are inserted for the Software AG IDL Compiler in the form `<parametername>=<value>`. If wildcards were used for the values, they are resolved to the actual values when the Custom Wrapper was called.

To start the Custom Wrapper in command-line mode

- See *Using EntireX in the Designer Command-line Mode* for the general syntax of the command line.

The command for the Custom Wrapper is `-xxx`, where `xxx` is the case-sensitive name of the Custom Wrapper. If the name contains blanks, use `-"xxx"`.

Example:

```
-NEW /Demo/example.idl
```

The result of the Custom Wrapper is written to Standard Out.

9 Using EntireX in the Designer Command-line Mode

- Command Line under Windows 62
- Command Line under Linux 63
- List of all Commands 63

For EntireX, the Designer can also be used from a command line. The command entered depends on your operating system.

- Under Windows, the command line is available with the starter *workbench.bat*.
- Under Linux, the command line is available with script *workbench.bsh*.
- Under both operating systems, the command line is also available with the command for Eclipse using the Java Runtime.

In all alternatives, a command is followed by a list of file names and a list of options. The file names may contain an asterisk (*) as a wildcard. The options are key-value pairs, where the key starts with a hyphen. The command selects the Wrapper or Extractor to use. The detailed options for each command are described in the respective Wrapper or Extractor section. Using `-help` as command lists all available commands with a short description. Using `-help <command>` lists the options of the command.

For a detailed description of each Wrapper or Extractor, see the documentation of this component. Throughout these detailed descriptions we use <*workbench*> as a general placeholder for the actual starter of the Designer. This can be `workbench.bat`, `workbench.bsh`, or the Eclipse starter.

Command Line under Windows

Enter the following command, replacing <*EntireX HOME*> with your EntireX installation directory:

```
<EntireX HOME>\bin\workbench.bat <command> [ <file> [ <file> ... ] ] [<options>]
```

This is the preferred method to start the EntireX design-time from the command line. Alternatively, you can use

```
"%ECLIPSE_HOME%\eclipsec" -application
com.softwareag.entirex.ide.eclipse.EntireXCommand -data %WORKSPACE% -nosplash
<command> [ <file> [ <file> ... ] ] [<options>]
```

where `ECLIPSE_HOME` is the Eclipse installation directory

`WORKSPACE` is the Eclipse workspace directory.

Command Line under Linux

Enter the following command:

```
/<Install_Dir>/EntireX/bin/workbench.bsh <command> [ <file> [ <file> ... ] ]
[<options>]
```

or

```
eclipse -vm $ECLIPSE_HOME/bin/ -application
com.softwareag.entirex.ide.eclipse.EntireXCommand -data $WORKSPACE -nosplash
<command> [ <file> [ <file> ... ] ] [<options>] -vmargs
-Dentirex.license=/<Install_Dir>/EntireX/common/conf/LKey/exx103.xml
```

where `ECLIPSE_HOME` is the Eclipse installation directory

`WORKSPACE` is the Eclipse workspace directory.

List of all Commands

Command	Description	Syntax / Examples
`-c:client`	Generate an RPC client from an IDL file.	*Using the C Wrapper in Command-line Mode*
`-c:server`	Generate an RPC server from an IDL file.	
`-cobol:client`	Generate a COBOL RPC client from an IDL file.	*Using the COBOL Wrapper in Command-line Mode*
`-cobol:server`	Generate a COBOL RPC server from an IDL file.	
`-dcom:generate`	Generate the DCOM Wrapper object(s) for the specified IDL file(s).	*Using the DCOM Wrapper in Command-line Mode*
`-deploy:cobol`	Deploy server-side mapping files (Designer file with extension .svm) to the specified environment. See ***Server Mapping Files for COBOL***.	***Command-line Mode***
`-extract:natural`	Extract the Natural sources or objects from a Natural RPC Server.	*Using the IDL Extractor for Natural in Command-line Mode*
`-extract:pli`	Extract the PL/I sources from an RPC Extractor Service.	*Using the IDL Extractor for PL/I in Command-line Mode*

Command	Description	Syntax / Examples
`-extract:wsdl`	Extract an IDL file and an XMM file from a Web service.	*Using the IDL Extractor for WSDL in Command-line Mode*
`-extract:xml`	Extract an IDL file and an XMM file from an XML Document.	*Using the IDL Extractor for XML Document in Command-line Mode*
`-extract:xsd`	Extract an IDL file and an XMM file from an XML Schema.	*Using the IDL Extractor for XML Schema in Command-line Mode*
`-help`	List the short description of all commands.	
`-java:all`	Generate all Java source files for the specified IDL file(s).	*Using the Java Wrapper in Command-line Mode*
`-java:allbeancompliant`	Generate all Java source files for the specified IDL file(s). The client object will be JavaBean-compliant.	
`-java:client`	Generate the Java client(s) for the specified IDL file(s).	
`-java:clientbeancompliant`	Generate the JavaBean-compliant Java client(s) for the specified IDL file(s).	
`-java:server`	Generate the Java server(s) for the specified IDL file(s).	
`-java:tester`	Generate the Java client(s) and tester(s) for the specified IDL file(s).	
`-list:natural`	List the Natural sources or objects from a Natural RPC Server.	*Using the IDL Extractor for Natural in Command-line Mode*
`-list:pli`	List the PL/I sources on an RPC Extractor Service.	*Using the IDL Extractor for PL/I in Command-line Mode*
`-map:soap`	Create SOAP-conformant XML mapping for all programs.	*Using the XML/SOAP Wrapper in Command-line Mode*
`-map:xmlattributes`	Create attribute-preferred XML mapping for all programs.	
`-map:xmlelements`	Create element-preferred XML mapping for all programs.	
`-map:xmlwithxsd`	Create element-preferred XML mapping with XML Schema for all programs.	
`-natural:client`	Generate Natural RPC client from the specified IDL file.	*Using the Natural Wrapper in Command-line Mode*
`-natural:server`	Generate Natural RPC server from the specified IDL file.	
`-pli:client`	Generate a PL/I RPC client from the specified IDL file.	*Using the PL/I Wrapper in Command-line Mode*

Command	Description	Syntax / Examples
`-pli:server`	Generate a PL/I RPC server from the specified IDL file.	
`-version`	Prints the version and exits.	
`-wsdl`	Generates WSDL for the specified IDL file(s).	*Using the Web Services Wrapper in Command-line Mode*
`-xml:sample`	Create sample XML documents for all programs.	*Using the XML/SOAP Wrapper in Command-line Mode*

10 Storing IDL Properties in an External File

When archiving an EntireX project with the export feature of Eclipse or the check-in feature of a revision control system such as Subversion or CVS, you can use an external properties file to store the properties of your IDL files. These properties are restored from this external file when the project is imported or checked out.

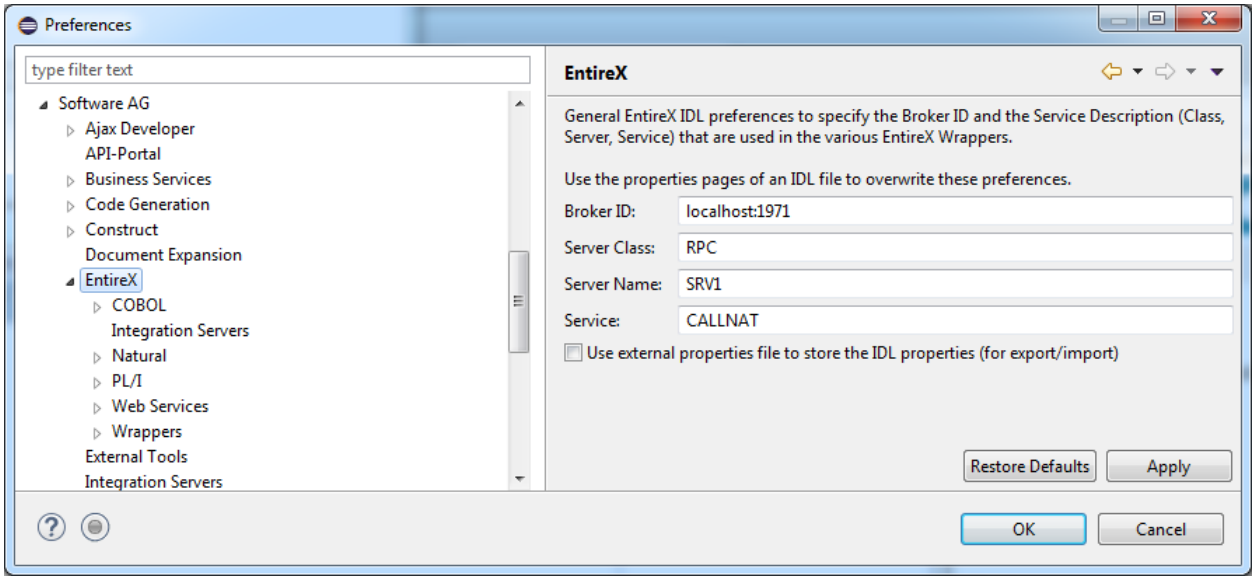
Note: This feature is not the default behavior and must be enabled for source and target environment as described below.

To enable the creation of external properties files

1 Start Designer and choose **Window** > **Preferences** > **Software AG** > **EntireX** and check the box **Use external properties file...**.

2 Modify the properties of your IDL file. If you make any changes to default values, these changes will be stored in a properties file with the name *<name>.idl.xml*.

When you archive your EntireX project using the Eclipse or a revision control system, make sure the external properties file is included for every IDL file. When you recreate your EntireX project in the same or a different environment, the properties of the IDL file are restored from the external properties file. In the target system, the box **Use external properties file...** must also be checked.


Preferences
type filter text
Software AG
Ajax Developer
API-Portal
Business Services
Code Generation
Construct
Document Expansion
EntireX
COBOL
Integration Servers
Natural
PL/I
Web Services
Wrappers
External Tools
Integration Servers
EntireX
General EntireX IDL preferences to specify the Broker ID and the Service Description (Class, Server, Service) that are used in the various EntireX Wrappers.
Use the properties pages of an IDL file to overwrite these preferences.
Broker ID:
localhost:1971
Server Class:
RPC
Server Name:
SRV1
Service:
CALLNAT
Use external properties file to store the IDL properties (for export/import)
Restore Defaults
Apply
OK
Cancel