webMethods EntireX

EntireX Workbench

Version 9.6

April 2014

Document ID: EXX-EEXXWORKBENCH-96-20140628

Table of Contents

EntireX Workbench v

1 Scope of the EntireX Workbench 1

- Workbench Editors 2
- EntireX Wrappers 3
- Software AG IDL Extractors 5
- Other Components 6

2 Supported File Types 7

- Input Files 8
- Output Files 8

3 EntireX IDL Tester 11

- Introduction 12
- Calling the IDL Tester 12
- Using a Broker with EntireX Security 15
- Using RPC Servers with Natural Security 15

4 Using EntireX Custom Wrappers 17

- Define EntireX Custom Wrappers 18
- Running a Custom Wrapper 25

5 Using the EntireX Workbench in Command-line Mode 27

- Command Line under Windows 28
- Command Line under Linux 29
- List of all Commands 29

6 Storing IDL Properties in an External File 33

EntireX Workbench

The EntireX Workbench is implemented and delivered as of a set of Eclipse plug-ins. It provides the EntireX development-time user interface in a commonly used environment. Among other capabilities, EntireX Workbench provides various wrappers, IDL extractors, editors and testers for generating and testing interface objects to be used in RPC communication between existing RPC servers and RPC client applications.

Scope	Lists the components of the EntireX Workbench, with links to more information.
Supported File Types	File types supported by EntireX Workbench.
IDL Tester	Using the EntireX IDL Tester.
Using EntireX Custom Wrappers	How to define your own template-based wrappers.
Command-line Mode	Using EntireX Workbench from a command line.
Storing IDL Properties	Using an external file to store IDL properties for export/import.

See also *Installing the EntireX Workbench Plug-ins in Standalone Mode* in the general installation documentation for how to install the EntireX Workbench Plug-ins. This document is applicable if you want to install the Workbench plug-ins in standalone mode, 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 Software AG Designer and then choose **Help** > **Help Contents**. General information on Eclipse can then be found under *Workbench User Guide*. When working with the Software AG Designer, the online help also provides help for the currently installed Software AG products; this can be found under *Software AG Designer Guides*.

1 Scope of the EntireX Workbench

- Workbench Editors 2
- EntireX Wrappers 3
- Software AG IDL Extractors 5
- Other Components 6

See also *Installing the EntireX Workbench Plug-ins in Standalone Mode* in the general installation documentation. These sections are applicable if you want to install the Workbench plug-ins in standalone mode, that is, not as part of the full webMethods EntireX installation.

Workbench 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 EntireX Workbench to generate interface objects based on a Software AG IDL file and additional wrapping properties. The following wrappers are provided:

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

C Wrapper
EntireX C Wrapper provides access to RPC-based components from C 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 Workbench 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.

Wrapper for EJB
The EntireX Wrapper for Enterprise JavaBeans enables Java-based components to access an EntireX RPC server using Enterprise JavaBeans. This is accomplished by mapping RPC server libraries to EJB stateful session beans. A remote procedure call will be mapped to a method of the corresponding Enterprise JavaBeans. The Java-based components can use the logic implemented in the libraries of the existing RPC servers, or they can implement new logic on the application server side, using parts of the old logic.

Java Wrapper
The EntireX Java Wrapper provides access to EntireX RPC-based components from Java applications. EntireX Java RPC enables users to 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.

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.

Web Services Wrapper
The EntireX Web Services Wrapper is a wizard that generates Web services from Software AG IDL, XML/SOAP mapping files or Natural subprogram files. The generated result is a Web service archive (.aar) that contains the relevant artifacts of the Web service such as an XML/SOAP mapping file (.xmm), WSDL file and additional configuration files. The Web service archive can be deployed for execution by the wizard or - in an extra deployment step - in a Web Services Stack with the EntireX XML/SOAP Listener runtime.

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 EntireX Workbench 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 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 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 XML Document
The Software AG IDL Extractor for XML Document generates a *Software AG IDL File* in the IDL Editor documentation 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 a *Software AG IDL File* in the IDL Editor documentation and a related XML mapping file (XMM) from given XML schema files.

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.

Other Components

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.

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 EntireX Workbench. It displays the status of the EntireX Default Broker and the active RPC Services registered to it.

2 Supported File Types

- Input Files 8
- Output Files 8

Input Files

The following table lists the file types that the EntireX Workbench 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	CVM file (client-side server mapping file) containing information on the IDL. Corresponds to a related IDL file with the same name, see *CVM File*.	
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	SVM file (server mapping file) containing information on the IDL. Corresponds to a related IDL file with the same name, see *Server Mapping File (SVM)*.	
wsdl	Web service description file.	IDL file, mapping file.
xml	XML document.	IDL file, 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, mapping file.

Output Files

The following table lists the file types generated by the EntireX Workbench.

File Type	Content	Generated by
aar	Service archive for Web Services Stack.	IDL file with XML mapping.
cvm	CVM file (client-side server mapping file).	Natural source.
idl	Software AG IDL for RPC.	Import XSD, import XML, import WSDL.
svm	SVM file (server mapping file).	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.

File Type	Content	Generated by
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.

3 EntireX IDL Tester

- Introduction 12
- Calling the IDL Tester 12
- Using a Broker with EntireX Security 15
- Using RPC Servers with Natural Security 15

Note: The Java Wrapper also provides a method of testing IDL files. This method was available in earlier EntireX versions and is useful if you need to generate test classes into a Java project. See *Using the IDL Tester* under *Using the Java Wrapper*.

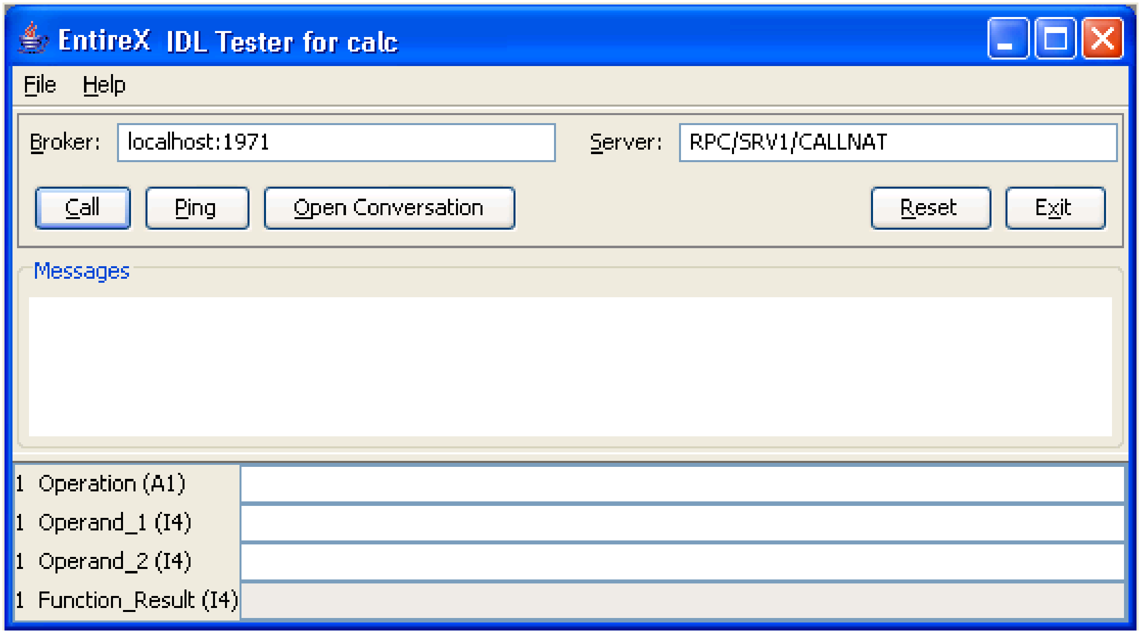
Introduction

The IDL Tester is an easy-to-use utility to check the accessibility of the Web service from the EntireX Broker via the XML/SOAP RPC server. It acts as an RPC client application, taking as input an IDL file. It verifies the mapping of the RPC data to the Web services data in the XML/SOAP mapping file.

Calling the IDL Tester

To call the IDL Tester

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



Broker: localhost:1971

Server: RPC/SRV1/CALLNAT

1 Operation (A1)	
1 Operand_1 (I4)	
1 Operand_2 (I4)	
1 Function_Result (I4)	

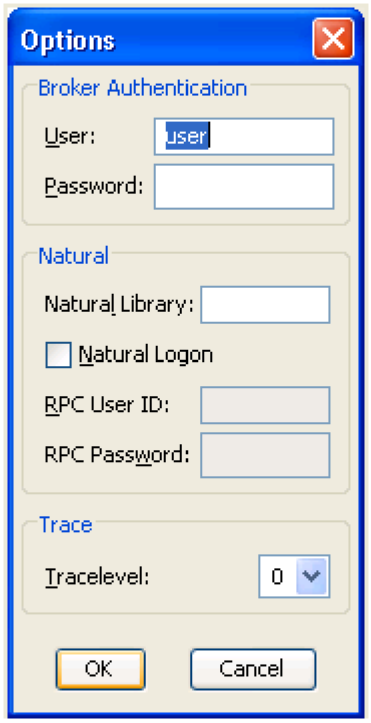
The dialog contains fields for Broker ID and server address and buttons, a message area for general output and error messages, and a parameter area with the input and output parameters. The fields for the input parameters have a white background and the fields for the output parameters are disabled and 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 setting user and password for the Broker Authentication. In addition, for communication with Natural RPC Server, the Natural library can be set and if Natural logon is enabled, user ID and password for RPC Authentication can be set.



User: user

Tracelevel: 0

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 a Broker with EntireX Security

When using a Broker which runs with EntireX Security, modify user ID and password accordingly.

Using RPC Servers with Natural Security

When using Natural Security, enable Natural logon and set the user ID and the password.

4 Using EntireX Custom Wrappers

- Define EntireX Custom Wrappers 18
- Running a Custom Wrapper 25

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

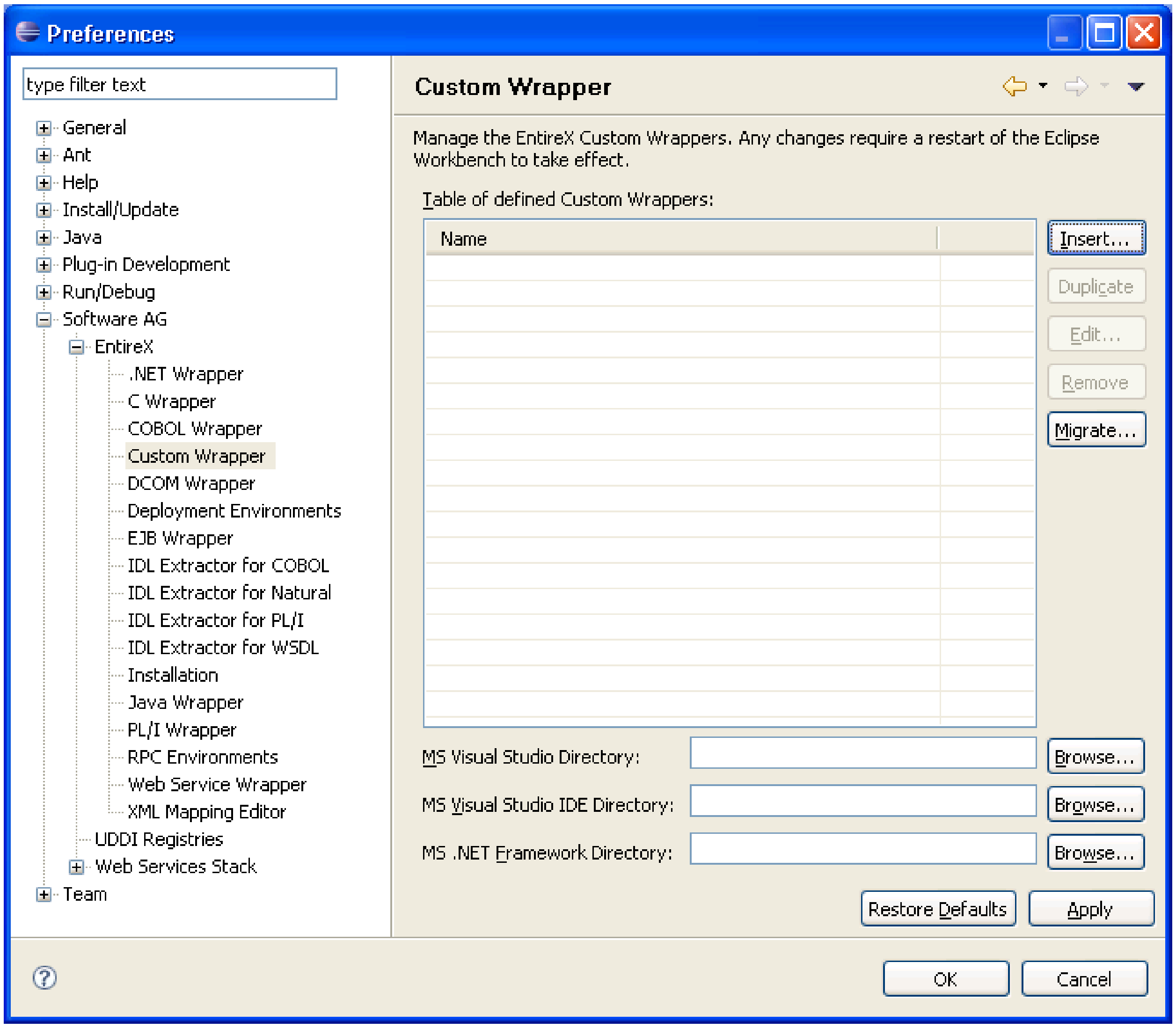
Naturally, existing *.plugin* files from former EntireX installations, called EntireX Workbench Plug-ins, can be migrated. The Custom Wrapper definitions are stored in the current Eclipse workspace and can be managed in the preferences. Any changes in the Custom Wrapper definitions require a restart of the Workbench 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


Preferences

type filter text

- General
- Ant
- Help
- Install/Update
- Java
- Plug-in Development
- Run/Debug
- Software AG
  - EntireX
    - .NET Wrapper
    - C Wrapper
    - COBOL Wrapper
    - Custom Wrapper
    - DCOM Wrapper
    - Deployment Environments
    - EJB Wrapper
    - IDL Extractor for COBOL
    - IDL Extractor for Natural
    - IDL Extractor for PL/I
    - IDL Extractor for WSDL
    - Installation
    - Java Wrapper
    - PL/I Wrapper
    - RPC Environments
    - Web Service Wrapper
    - XML Mapping Editor
  - UDDI Registries
  - Web Services Stack
- Team

Custom Wrapper

Manage the EntireX Custom Wrappers. Any changes require a restart of the Eclipse Workbench to take effect.

Table of defined Custom Wrappers:

Name

MS Visual Studio Directory:

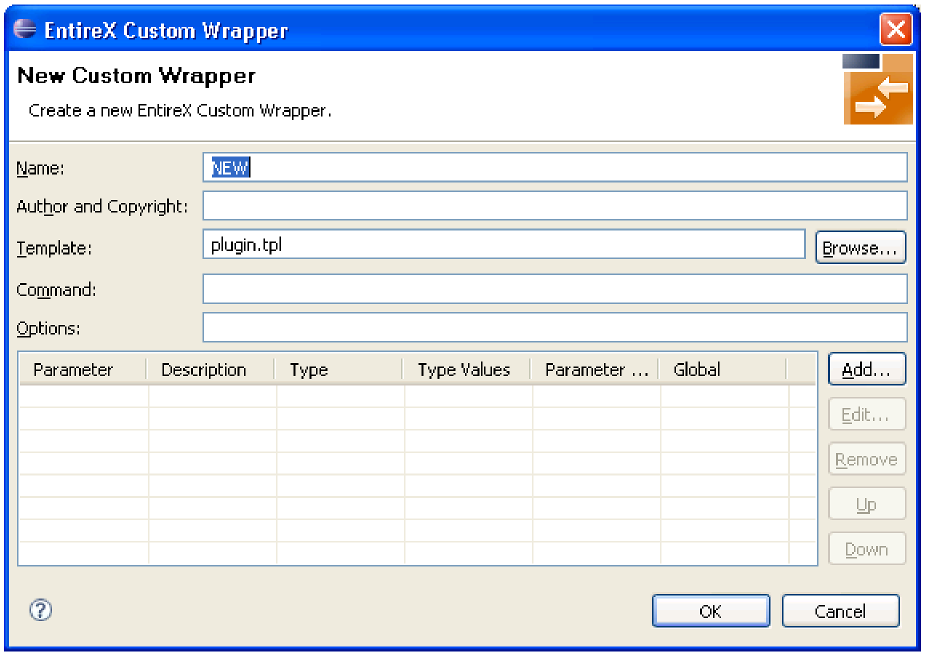
MS Visual Studio IDE Directory:

MS .NET Framework Directory:

Creating a New Custom Wrapper

To create a new Custom Wrapper

1 Choose **Insert...**



Name: NEW
Author and Copyright:
Template: plugin.tpl
Command:
Options:

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.

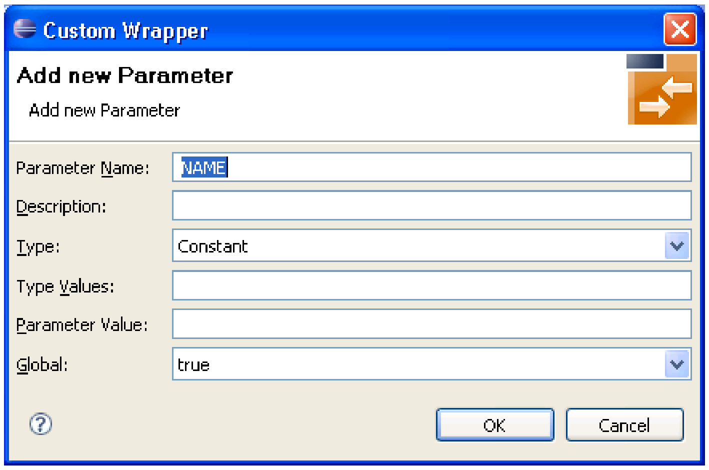
Field	Description
Name	Used as the Custom Wrapper menu item text.
Author and copyright	Important if you want to share your Custom Wrapper with other users.
Template	Name of the template file to be used by the *Software AG IDL Compiler* in the IDL Editor documentation. Use a fully qualified file name.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


Custom Wrapper

Add new Parameter

Add new Parameter

Parameter Name: NAME

Description:

Type: Constant

Type Values:

Parameter Value:

Global: true

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.
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 EntireX Workbench.	C:\Examples\example.idl
`%idlname`	Get the current IDL file (just the file name with extension, no path) as selected in the EntireX Workbench.	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 EntireX Workbench.	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

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 the **Duplicate** button instead of the **Insert...** button.

   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*.

Edit an existing Custom Wrapper

1. Select the Custom Wrapper in the list.
2. Proceed as described in *Creating a New Custom Wrapper* and select the **Edit...** button instead of the **Insert...** button
3. Modify the entries as described in *Creating a New Custom Wrapper*.

Remove an existing Custom Wrapper

1. Select the Custom Wrapper in the list.
2. Use the **Remove** button

Migrate an existing *.plugin* File from a previous EntireX Installation

1. Proceed as described in *Creating a New Custom Wrapper* and select the **Migrate...** button instead of the **Insert...** button.
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 **Generate ... from Software AG IDL**, choose **Via Template** and select the desired item (for example, "Insert...").

When the Custom Wrapper is started, the EntireX Workbench 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

- See *Using the EntireX Workbench in 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.

5 Using the EntireX Workbench in Command-line Mode

- Command Line under Windows 28
- Command Line under Linux 29
- List of all Commands 29

The EntireX Workbench 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 Workbench. 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 Workbench. Alternatively, you can start the EntireX Workbench with

```
"%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:

```
/opt/softwareag/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=/opt/softwareag/EntireX/common/conf/LKey/exx96.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*
`-ejb:generate`	Generate the EJB sources for the specified IDL file(s).	*Using the Wrapper for EJB in Command-line Mode*
`-deploy:cobol`	Deploy server mapping files (SVMs) to the specified environment.	*Server Mapping Deployment in Command-line Mode*
`-extract:natural`	Extract the Natural objects from a Natural RPC Server.	*Using the IDL Extractor for Natural in Command-line Mode*
`-extract:pli`	Extract the PL/I source objects 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 objects from a Natural RPC Server.	*Using the IDL Extractor for Natural in Command-line Mode*
`-list:pli`	List the PL/I source objects 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 source files 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*

6 Storing IDL Properties in an External File

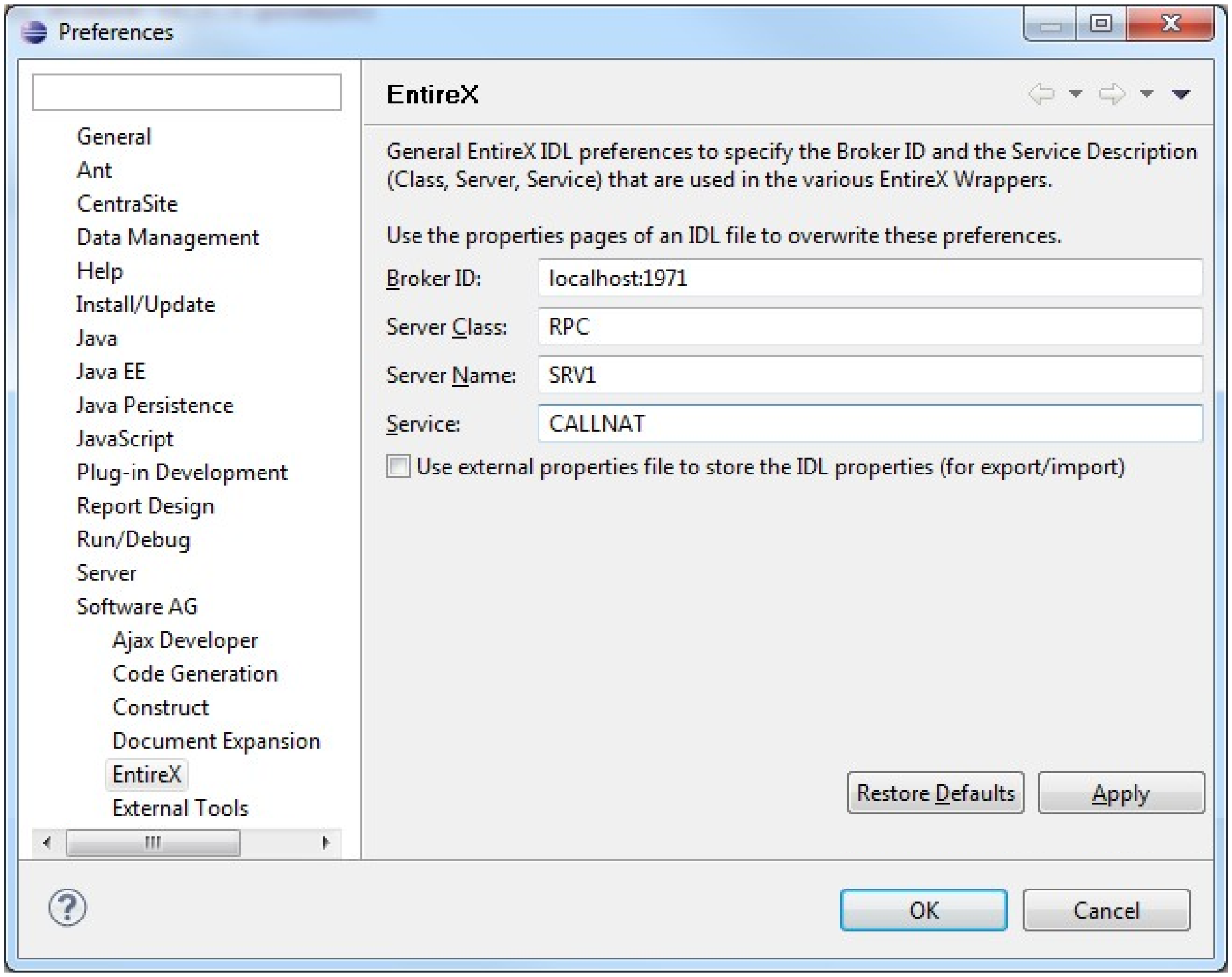
When archiving an EntireX project with the export feature of Eclipse or the check-in feature of a revision control system such as Subversion or CVS, you can use an external properties file to store the properties of your IDL files. These properties are restored from this external file when the project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



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)