

## **webMethods EntireX**

**Software AG IDL Extractor for XML Schema**

Version 9.6

April 2014

This document applies to webMethods EntireX Version 9.6.

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

Copyright © 1997-2014 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://documentation.softwareag.com/legal/>.

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://documentation.softwareag.com/legal/> 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 and license terms, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". This document is part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).

**Document ID: EXX-EEXXMLSCHEMAEXTRACTOR-96-20140628**

## Table of Contents

1 Introduction to the Software AG IDL Extractor for XML Schema .....	1
2 Using the Software AG IDL Extractor for XML Schema .....	3
Step 1: Start the IDL Extractor for XML Schema .....	4
Step 2: Select a Source .....	5
Step 3a: Specify XML File .....	6
Step 3b: Specify XML File URL .....	7
Step 4: Specify Output Files .....	9
Step 5: Specify Options for Target Programming Language .....	9
Extraction Result .....	13
3 Using the IDL Extractor for XML Schema in Command-line Mode .....	15
4 XML Schema to IDL Mapping .....	17
Extracting IDL from XML Schema .....	18
XML Schema Parameter Mapping .....	18
Extracting the Name for the IDL Library .....	19
Extracting the Name for the IDL Program .....	19



# 1 Introduction to the Software AG IDL Extractor for XML

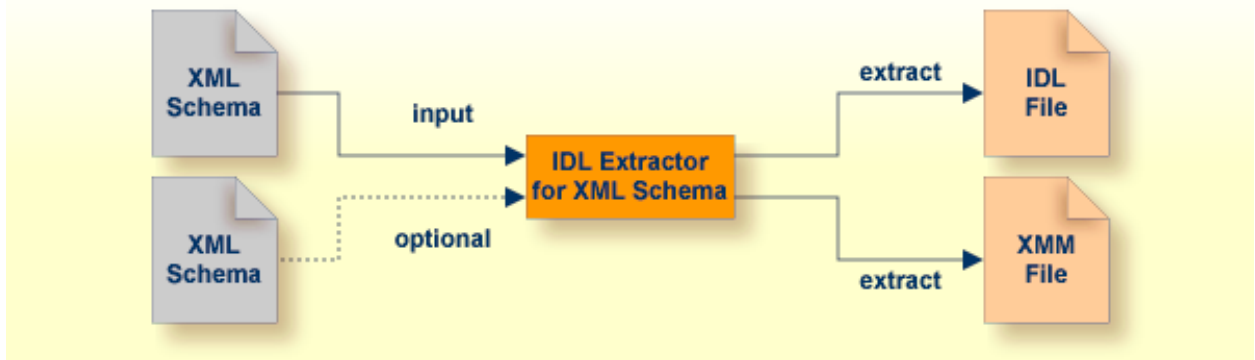
## Schema

---

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.

The *EntireX Workbench* provides a wizard to collect all necessary input. You can extract from either

- a remote location using an URL (see [Step 2: Select a Source](#) for supported protocols), or
- directly from your Eclipse workspace.



For the direction OUT - that is, the reply from the server - an optional XML Schema file can be specified to extract the parameters returned. See [Step 3a: Specify XML File](#) for more information.



## 2 Using the Software AG IDL Extractor for XML Schema

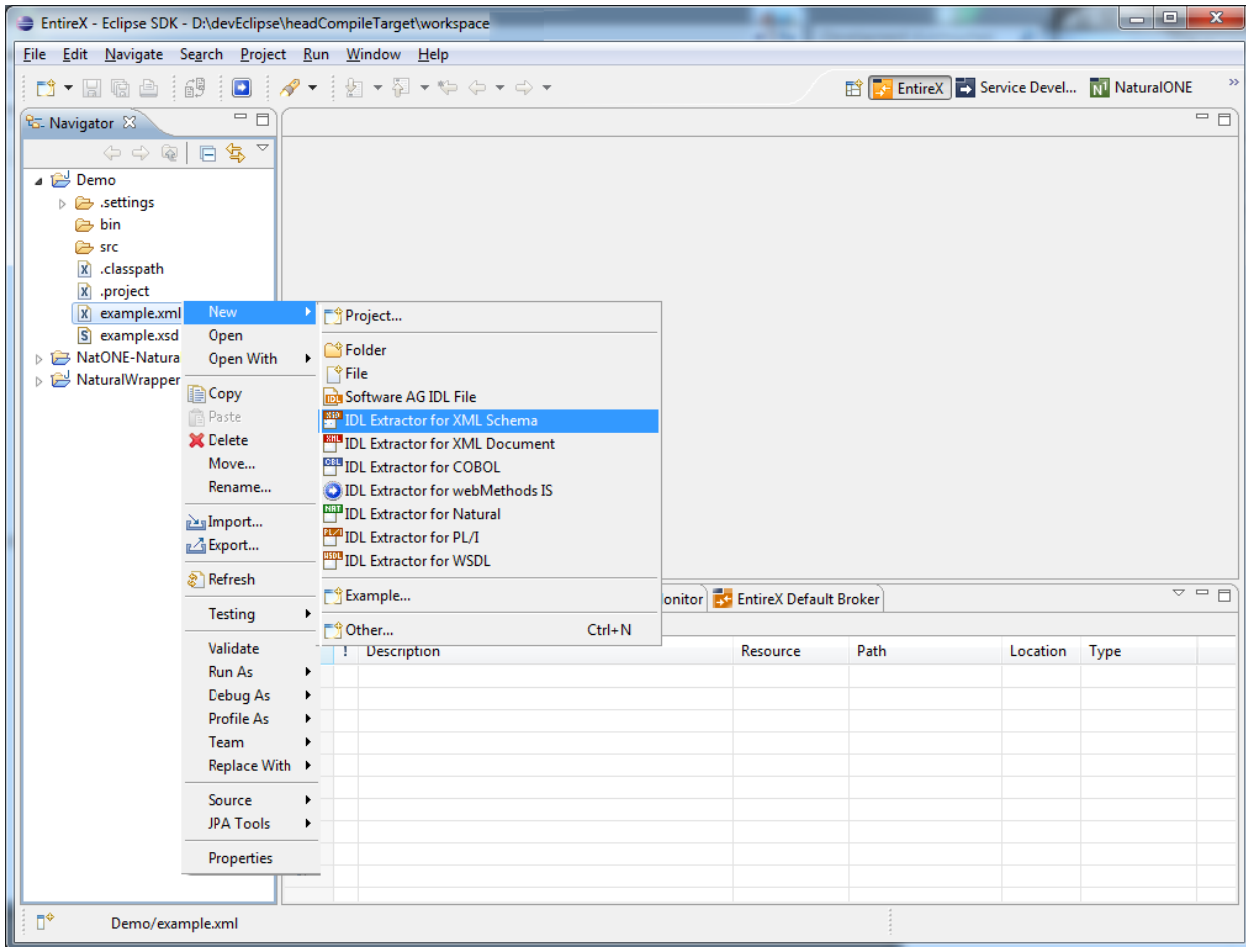
---

▪ Step 1: Start the IDL Extractor for XML Schema .....	4
▪ Step 2: Select a Source .....	5
▪ Step 3a: Specify XML File .....	6
▪ Step 3b: Specify XML File URL .....	7
▪ Step 4: Specify Output Files .....	9
▪ Step 5: Specify Options for Target Programming Language .....	9
▪ Extraction Result .....	13

**Caution:** If you modify the imported IDL file, do this only in the XML Mapping Editor to ensure the correct dependencies between the IDL and the related XMM file.

## Step 1: Start the IDL Extractor for XML Schema

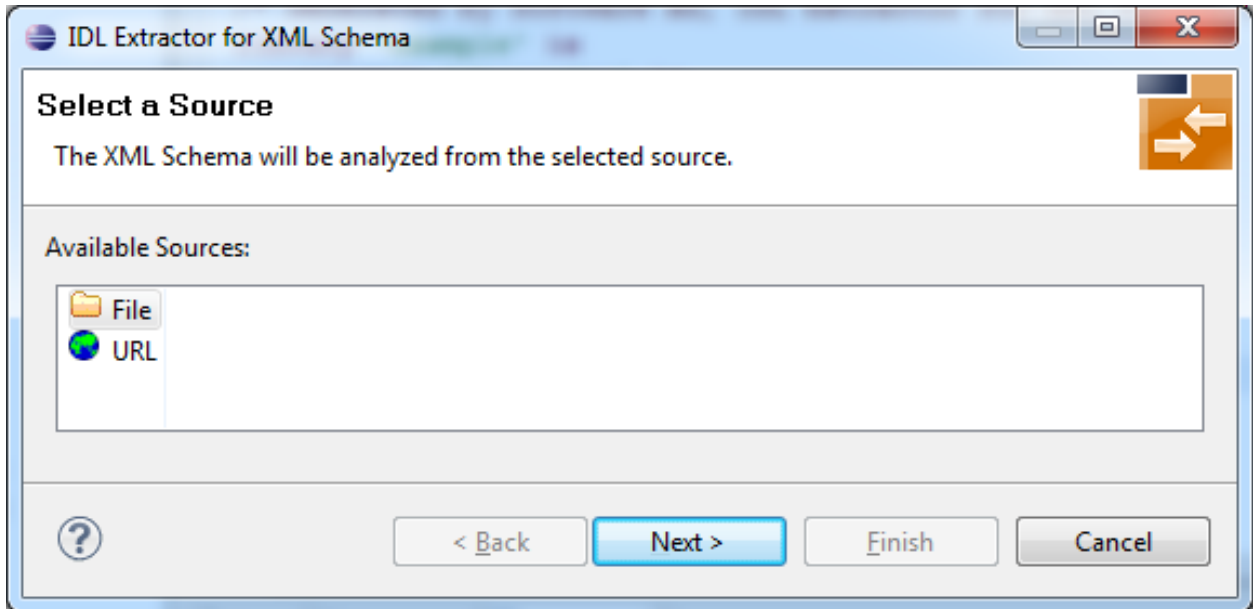
Start the IDL Extractor for XML Schema as any other eclipse New wizard:





## Step 2: Select a Source

Depending on the location of the XML Schema to analyze, choose **File** or **URL**:



### ■ File

If the XML Schema source file to be extracted is available in your workspace and you have selected it, the file location will be entered in the wizard automatically in the next [Step 3a: Specify XML File](#).



**Note:** Nested imports for XML Schema should be addressed in the same way as the base document. If those nested imports are addressed relatively, all of the documents must be in your file system.

### ■ URL

Continue with [Step 3b: Specify XML File URL](#).



#### Notes:

1. The supported URL protocols are FILE, FTP, HTTP, HTTPS and JAR, for example

```
http://host/myservice?XSD
```

2. If the connection is over HTTPS, you need to set up HTTPS in Software AG Designer:

Define `trustStore` in Designer, for example with the following lines in file `eclipse.ini`

```
-Djavax.net.ssl.trustStore=<path to keystore>  
-Djavax.net.ssl.trustStorePassword=<keystore password>
```

If hostname verification for certification is to be disabled, also add the line:

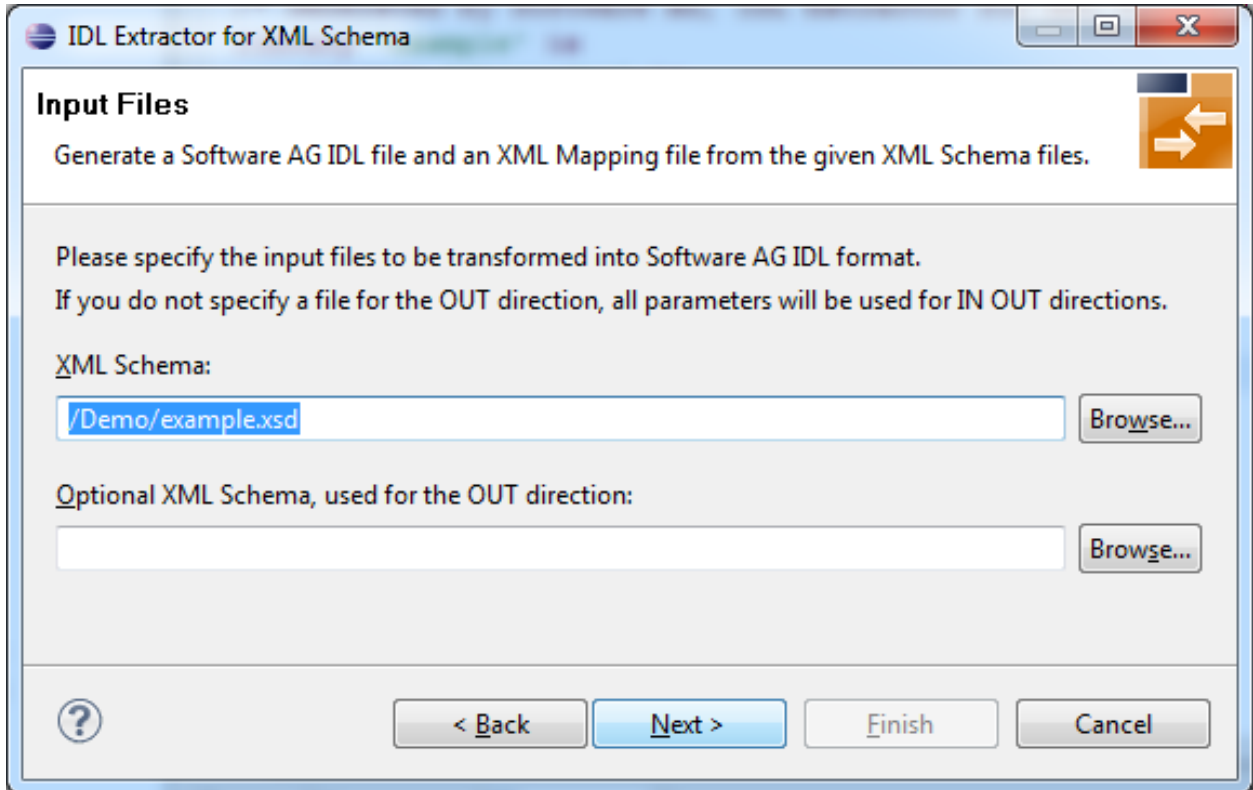
```
-Dcom.softwareag.entirex.ssl.hostnameverify=false
```

## Step 3a: Specify XML File

---

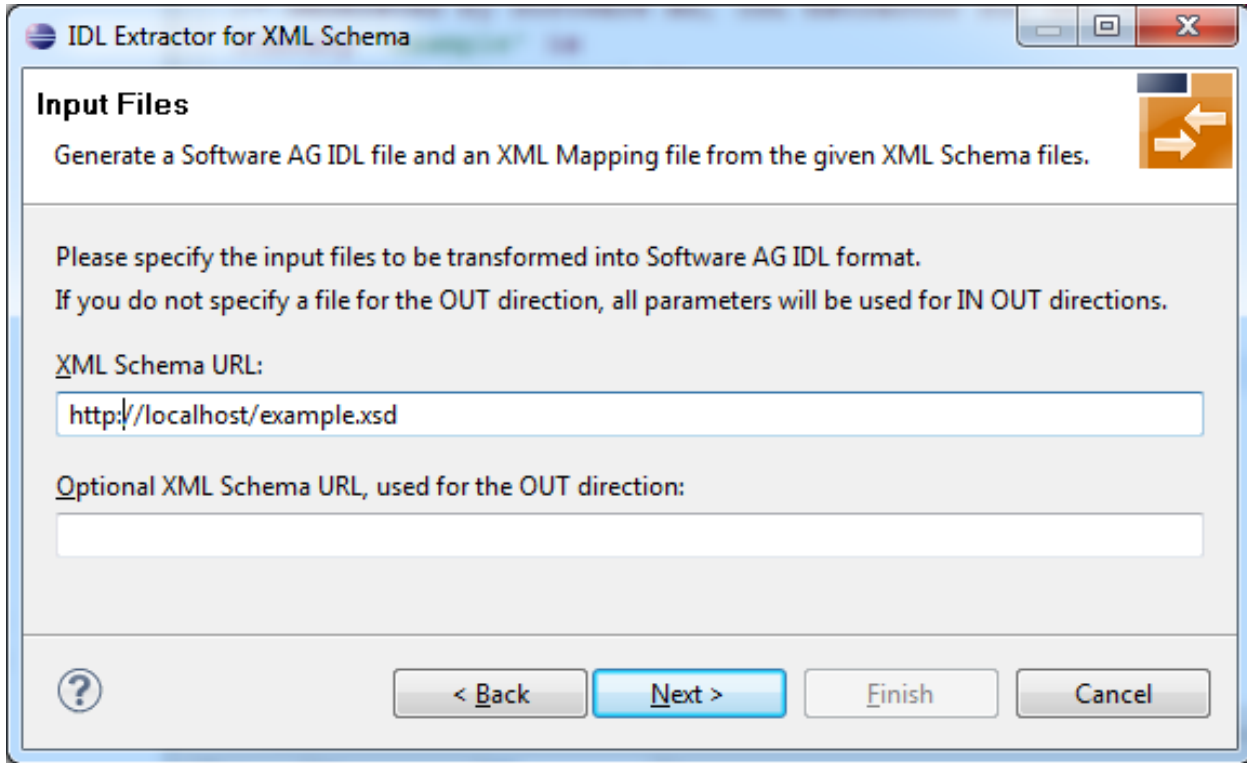
If you selected the XML Schema source file before you started the wizard, the file location is already present. Enter or browse for the XML Schema source file. Continue with [Step 4: Specify Output Files](#).

For the direction `OUT` - that is, the reply from the server - an optional XML Schema file can be specified to extract the parameters of the interface. If no additional XML Schema file is given, parameters for direction `IN` and `OUT` are the same and extracted from the same input XML Schema file. The resulting *Software AG IDL File* in the IDL Editor documentation contains all the parameters of the interface as IDL parameters (see `parameter-data-definition` under *Software AG IDL Grammar* in the *IDL Editor* documentation) with its `direction` (see `attribute-list` under *Software AG IDL Grammar* in the *IDL Editor* documentation).

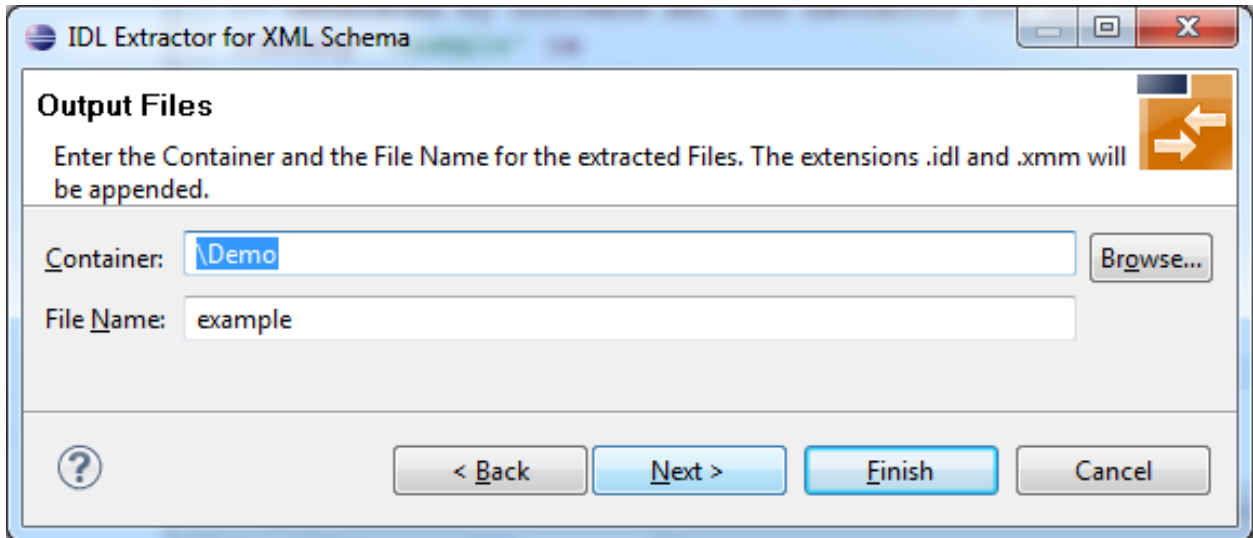


### Step 3b: Specify XML File URL

Enter the URL for the XML Schema source file. You can specify an optional XML Schema file for the IDL direction `OUT`. See [Step 3a: Specify XML File](#) for more information.



## Step 4: Specify Output Files

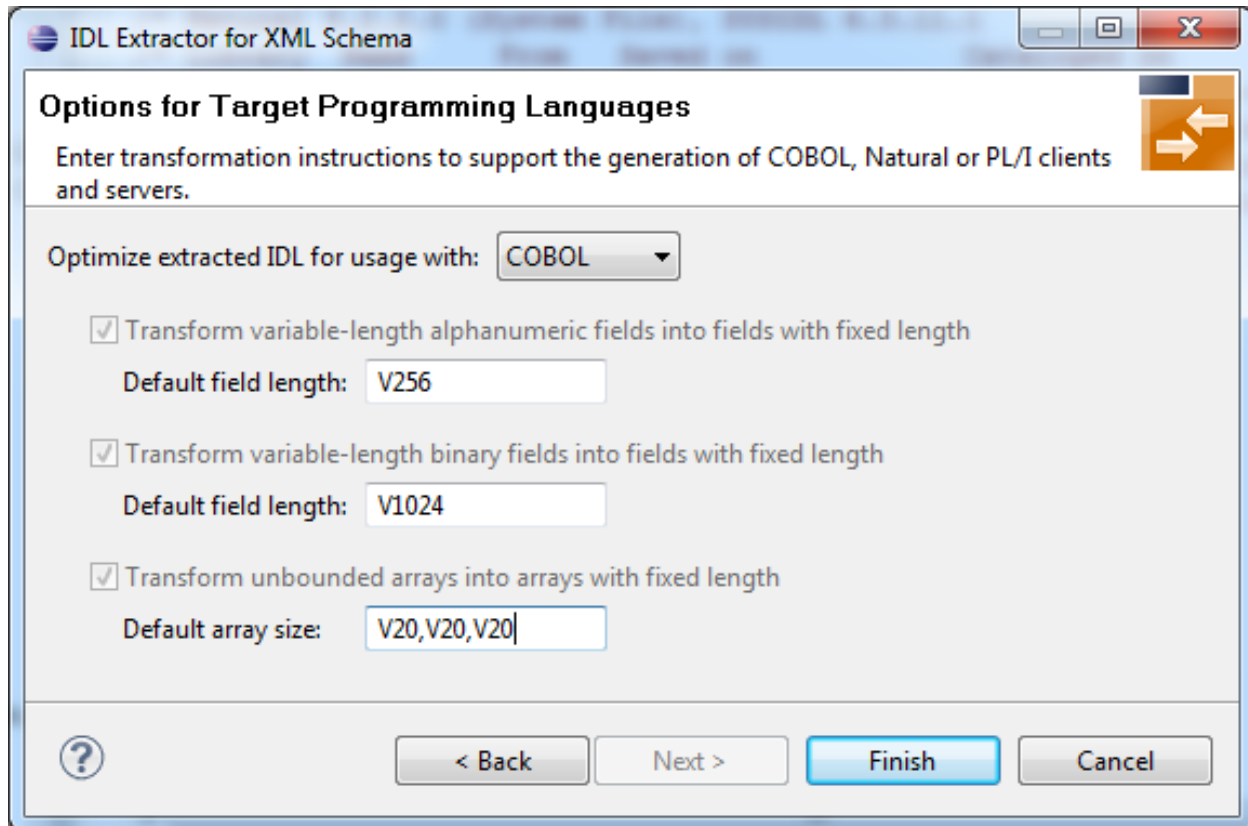


The screenshot shows a dialog box titled "IDL Extractor for XML Schema" with a standard Windows window border. The main heading is "Output Files". Below the heading, there is a text instruction: "Enter the Container and the File Name for the extracted Files. The extensions .idl and .xmm will be appended." To the right of this text is a small icon of two arrows pointing right. Below the instruction, there are two input fields: "Container:" with the text "\Demo" and a "Browse..." button to its right; and "File Name:" with the text "example". At the bottom of the dialog, there is a help icon (a question mark in a circle) on the left, and four buttons: "< Back", "Next >", "Finish", and "Cancel".

Select the Container where the IDL file will be stored. Enter the name of the new IDL file and the related XML mapping file.

## Step 5: Specify Options for Target Programming Language

The **Options for Target Programming Language** page allows you to specify transformation rules for variable-length fields and unbounded arrays. This is required if you later use the COBOL Wrapper or PL/I Wrapper with the extracted IDL – otherwise COBOL or PL/I wrapping is not possible. If you later use the Natural Wrapper, transformation rules are optional. If they are used, the interface from a Natural point of view is more legacy-like, easier to use but with restrictions.



With the transformation rules, you define default (maximum) lengths and sizes depending on the originating data types on the XML side. If you need different (maximum) lengths and sizes for fields with the same data type, use the XML Mapping Editor. See *Using the XML Mapping Editor*

**Caution:** If you modify the imported IDL file, do this only in the XML Mapping Editor to ensure the correct dependencies between the IDL and the related XMM file.

Depending on the target programming language of your scenario, the available/possible transformation rules differ. Use the combo-box and choose the target programming language:

- COBOL
- Natural
- PL/I Client
- PL/I Server

- Other

## COBOL

For generation of clients and servers with the *COBOL Wrapper*.

Variable-length fields and unbounded arrays with unlimited number of elements are not directly supported by COBOL. There are two possibilities to specify options:

- **Transform to Fixed-length COBOL Fields and Tables**

Variable-length fields on the XML side are mapped to fixed-length COBOL data items, that is, they will always be padded (alphanumeric with trailing blanks; binary with x00). Unbounded arrays on the XML side are mapped to fixed-size COBOL tables, see *Tables with Fixed Size* under *COBOL to IDL Mapping* in the IDL Extractor for COBOL documentation. This means they will always be filled up to the maximum number of elements. To use this possibility, enter the length or size to define the restriction, for example 256, 1024 or 20.

- **Limit Variable-length Fields and Unbounded Arrays to a Maximum**

For variable-length fields, EntireX provides a possibility to transform them into variable-length fields with a maximum length. See *IDL Data Types* under *Software AG IDL File* in the IDL Editor documentation, *AVnumber* and *BVnumber* under column *Type* and *Length*. In this case the variable-length fields are also mapped to fixed-length COBOL data items, but they will be trimmed (alphanumeric with blank, binary with x00) on the COBOL side. Unbounded arrays with a maximum are directly supported in COBOL in the form of COBOL tables with the `OCCURS DEPENDING ON` clause, see *Tables with Variable Size - DEPENDING ON Clause*. Only filled elements are transferred. In this case the RPC message size is reduced compared with the alternative *Transform to Fixed-length COBOL Fields and Tables* above. To use this possibility, enter a leading V-character before the limited length or limited size of unbounded arrays, such as V256, V1024 or V20.

## Natural

For generation of clients and servers with the *Natural Wrapper*.

Variable-length fields and unbounded arrays with unlimited number of elements are directly supported by Natural. As an alternative, EntireX also provides the possibility to transform to a more legacy-like interface with fixed length.

- **Transform to Fixed-length Fields and Fixed-size Arrays on the Natural Side**

Variable-length fields on the XML side are mapped to fixed-length Natural data types, that is, they will always be padded (alphanumeric with trailing blanks; binary with x00). Unbounded arrays on the XML side are mapped to fixed-length Natural arrays, that is, they will always be filled up to the maximum number of elements. Using this possibility you benefit from easier and simpler Natural programming. To use this possibility, check the check boxes and enter the restricted length for variable-length alphanumeric fields, such as 253, variable-length binary fields such as 126, and the restricted size, for example 20,20,20 for unbounded arrays.

■ **Transform to Variable-length Fields and Variable-size Arrays on the Natural Side**

Variable-length fields on the XML side are mapped to Natural DYNAMIC data types. No padding occurs on the Natural side. Unbounded arrays on the XML side are mapped to Natural X-Arrays. Only filled elements are transferred. In this case the RPC message size is reduced compared with the alternative *Transform to Fixed-length Fields and Fixed-size Arrays on the Natural Side* above. To use this possibility, uncheck the check boxes.

**PL/I Client**

For generation of clients with the *PL/I Wrapper*. The following possibilities exist in scenarios with PL/I clients:

■ **Transform to Fixed-length Fields and Arrays**

Variable-length fields on the XML side are mapped to fixed-length PL/I data items, that is, they will always be padded (alphanumeric with trailing blanks; binary with x00). Unbounded arrays on the XML side are mapped to fixed-size PL/I arrays, see *Arrays* under *PL/I to IDL Mapping*. This means they will always be filled up to the maximum number of elements. To use this possibility, enter the length or size to define the restriction, for example 256, 1024 or 20.

■ **Limit Variable-length Fields to a Maximum**

As an alternative, variable-length fields can be mapped to PL/I data type with the attribute VARYING. See also *IDL Data Types* under *Software AG IDL File* in the IDL Editor documentation AVnumber and BVnumber under column Type and Length. In this case no padding occurs on the PL/I side. To use this possibility, enter a leading V-character before the limited length, such as V256 or V1024.



**Note:** This alternative does not exist for unbounded arrays.

**PL/I Server**

For generation of servers with the *PL/I Wrapper*. The following possibilities exist in scenarios with PL/I servers:

■ **Transform to Fixed-length Fields and Arrays**

Variable-length fields on the XML side are mapped to fixed-length PL/I data items, that is, they will always be padded (alphanumeric with trailing blanks; binary with x00). Unbounded arrays on the XML side are mapped to fixed-size PL/I arrays, see *Arrays* under *PL/I to IDL Mapping* in the IDL Extractor for PL/I documentation. This means they will always be filled up to the maximum number of elements. To use this possibility, enter the length or size to define the restriction, for example 256, 1024 or 20.



### ■ Limit Variable-length Fields to a Maximum

As an alternative, variable-length fields can be mapped to PL/I data type with the attribute `VARYING`. See also *IDL Data Types* under *Software AG IDL File* in the IDL Editor documentation, `AVnumber` and `BVnumber` under column `Type` and `Length`. In this case no padding occurs on the PL/I side. To use this possibility, enter a leading `V`-character before the limited length, such as `V256` or `V1024`.



**Note:** This alternative does not exist for unbounded arrays.

### ■ Transform to Variable-size Arrays on the PL/I Side

As an alternative for unbounded arrays on the XML side, they can be mapped to PL/I arrays using `(*,*,*)` notation. Only filled elements are transferred. Note that PL/I does not allow resizing of these data types and arrays. In this case the RPC message size is reduced compared with the first alternative *Transform to Fixed-length PL/I Fields and Arrays* above. To use this possibility, uncheck the check box.



**Note:** This alternative does not exist for variable-length fields.

## Other

If you later use wrappers other than the COBOL Wrapper, Natural Wrapper or PL/I Wrapper, no transformation rules are required. Variable-length fields and unbounded arrays are extracted as is; there are no restrictions regarding data length that can be transferred in variable-length fields and the number of elements that can be transferred in unbounded arrays.

Press **Finish** to start extraction.

## Extraction Result

When the operation is completed, the IDL file is opened with the *Software AG IDL Editor*.

If the XML Schema source files to extract from contain parameters that cannot be mapped to IDL parameters, an IDL file with incorrect IDL syntax is created. The unsupported parameters lead to IDL parameters of data type `Error`, which is not supported. In the **Problems View** you get a marker for the first error in the IDL file.



# 3 Using the IDL Extractor for XML Schema in Command-line

## Mode

---

See *Using the EntireX Workbench in Command-line Mode* for the general command-line syntax. The table below shows the command-line option for the IDL Extractor for XML Schema.

Task	Command	Option	Description
Extract an IDL file and an XMM file from an XML Schema.	-extract:xsd	-help	Display this usage message.
		-out	The XML Schema for the OUT direction, whereby the first XML Schema will be interpreted as IN direction, instead of INOUT direction.
		-project	Name of the project or subfolder where the IDL and XMM files are stored.

### Example

```
<workbench> -extract:xsd /Demo/example.xsd
```

where *<workbench>* is a placeholder for the actual Workbench starter as described under *Using the EntireX Workbench in Command-line Mode*.

Status and processing messages are written to standard output (stdout), which is normally set to the executing shell window.



# 4 XML Schema to IDL Mapping

---

- Extracting IDL from XML Schema ..... 18
- XML Schema Parameter Mapping ..... 18
- Extracting the Name for the IDL Library ..... 19
- Extracting the Name for the IDL Program ..... 19

## Extracting IDL from XML Schema

The IDL Extractor for XML Schema interprets the first root definition as the root element of the document. Optionally, you can enter a second XML Schema file for the OUT direction (the first one will be used as the IN direction instead of the IN-OUT direction). For the parameter mapping see table *XML Schema Parameter Mapping* below. The XML mapping and IDL parameter directions are IN (Request) and OUT (Response), no Fault mapping trees will be created, see *XML Schema Parser Standards Conformance*.

## XML Schema Parameter Mapping

WSDL / XML Schema	XMM	Software AG IDL
binary, base64Binary	binary	BV (or BVn or Bn) <sup>(3)</sup>
hexBinary <sup>(1)</sup>	binary	BV (or BVn or Bn) <sup>(3)</sup>
boolean	boolean	L
date	date:yyyy-MM-dd <sup>(2)</sup>	D
float	float	F4
double	float	F8
byte, unsignedByte	integer	I1
short, unsignedShort	integer	I2
int, unsignedInt	integer	I4
integer, positiveInteger, nonPositiveInteger, negativeInteger, nonNegativeInteger	number	N29.0
decimal, number	number	N22.7
long, unsignedLong	number	N19.0
time	dateTime:HH:mm:ss <sup>(2)</sup>	T
dateTime	dateTime:yyyy-MM-dd'T'HH:mm:ss <sup>(2)</sup>	T
gYearMonth	string	A8
gDay, gYear	string	A11
gMonth	string	A12
gMonthDay	string	A13
string (and all types not listed here)	string	AV (or AVn or An) <sup>(3)</sup>



### Notes:

1. The hexBinary format is not supported by the XML/SOAP Runtime.

2. Edit the `date` and `dateTime` patterns manually to match the formats of the original documents.

**Example:** `<myTime xsi:type="xsd:date">11:08:23+01:00</myTime> --> dateTime:HH:mm:ss'+01:00 ' --> T`



**Note:** The `+01:00` is not supported by IDL (EntireX RPC protocol).

3. Mapped according to specified transformation rules. See [Step 5: Specify Options for Target Programming Language](#).

## Extracting the Name for the IDL Library

---

The IDL library name (see `library-definition` under *Software AG IDL Grammar* in the *IDL Editor* documentation) will be extracted from the source file name of the XML Schema (SOAP document).

## Extracting the Name for the IDL Program

---

The IDL program name (see `program-definition` under *Software AG IDL Grammar* in the *IDL Editor* documentation) will be extracted from each `<xsd:element>` (`xmlns:xsd="http://www.w3.org/2001/XMLSchema"`) level-one definition. Multiple IDL programs are possible.

