webMethods EntireX

Software AG IDL Extractor for XML Document

Version 9.6

April 2014

webMethods EntireX

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-EEXXXMLDOCEXTRACTOR-96-20140628

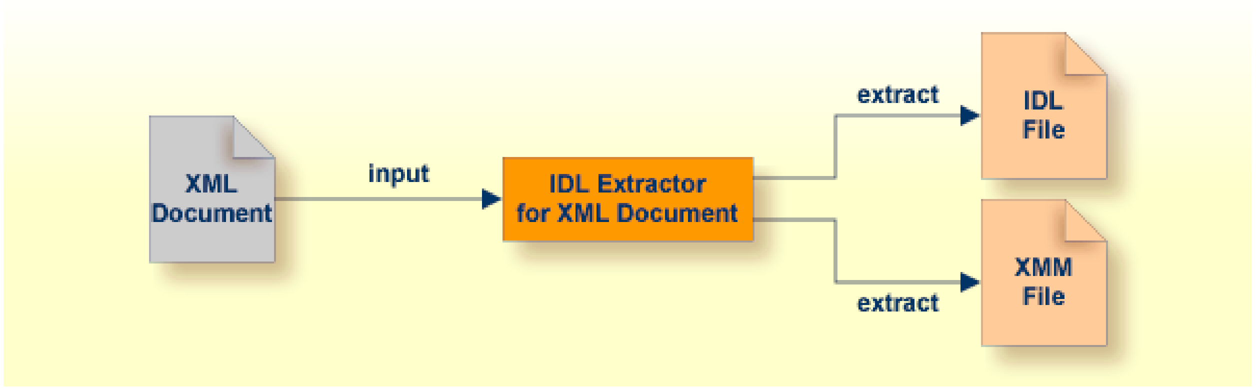
Table of Contents

1 Introduction to the Software AG 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.

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

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

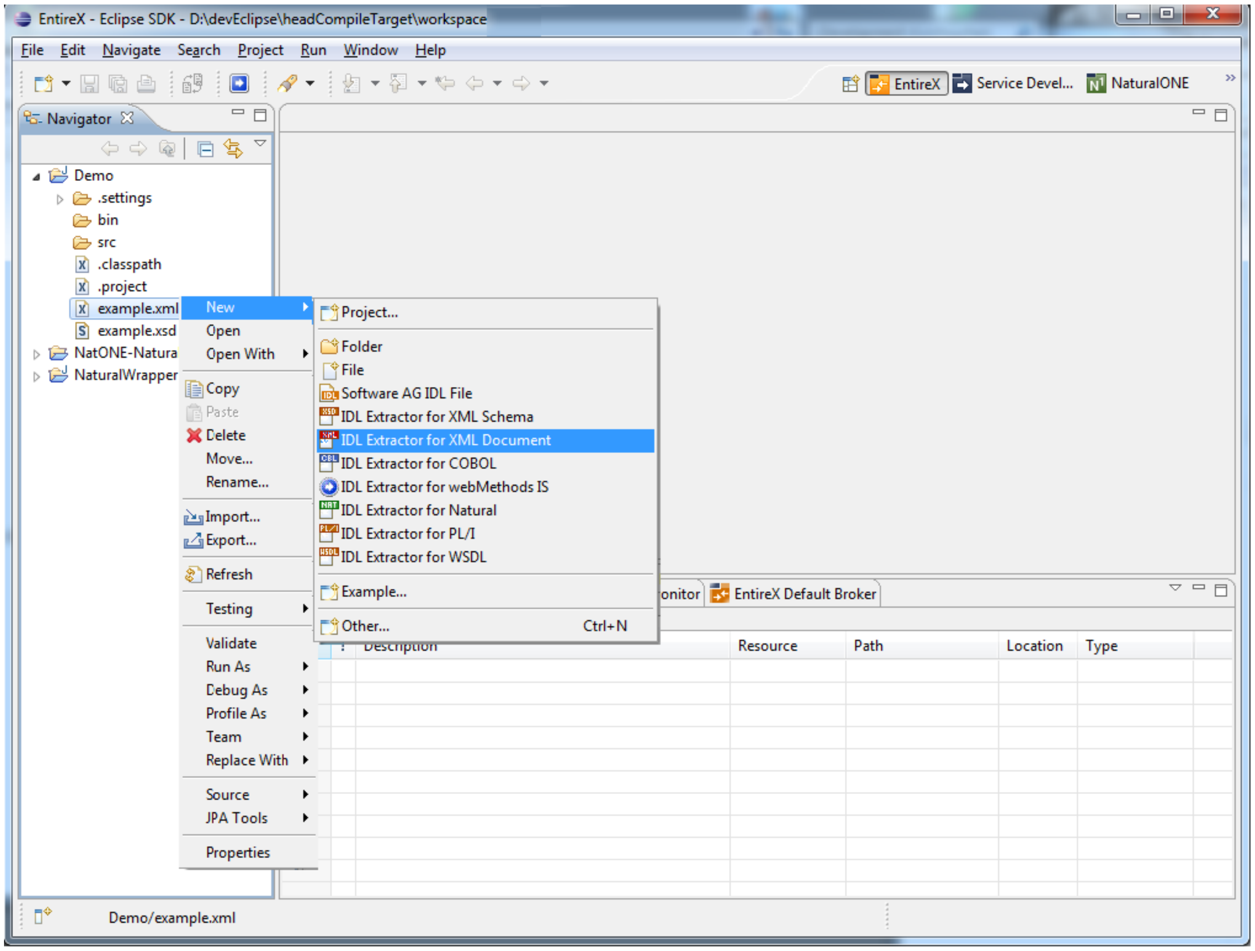


2 Using the Software AG IDL Extractor for XML Document

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.

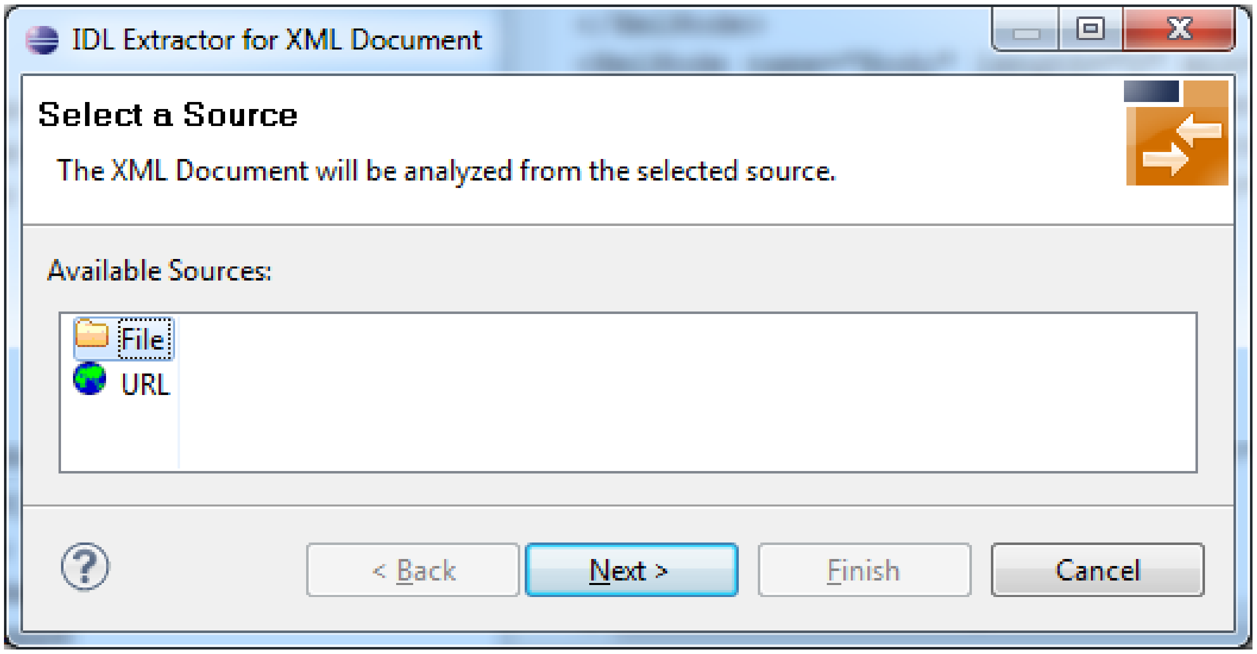
Step 1: Start the IDL Extractor for XML Document

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



Step 2: Select a Source

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


IDL Extractor for XML Document
Select a Source
The XML Document will be analyzed from the selected source.
Available Sources:
File
URL
< Back
Next >
Finish
Cancel

- **File**
  If the XML Document 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*.
- **URL**
  Continue with *Step 3b: Specify XML File URL*.

Notes:

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

   ```
   jar:file:/C:test.jar!/Test.xml
   ```

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*

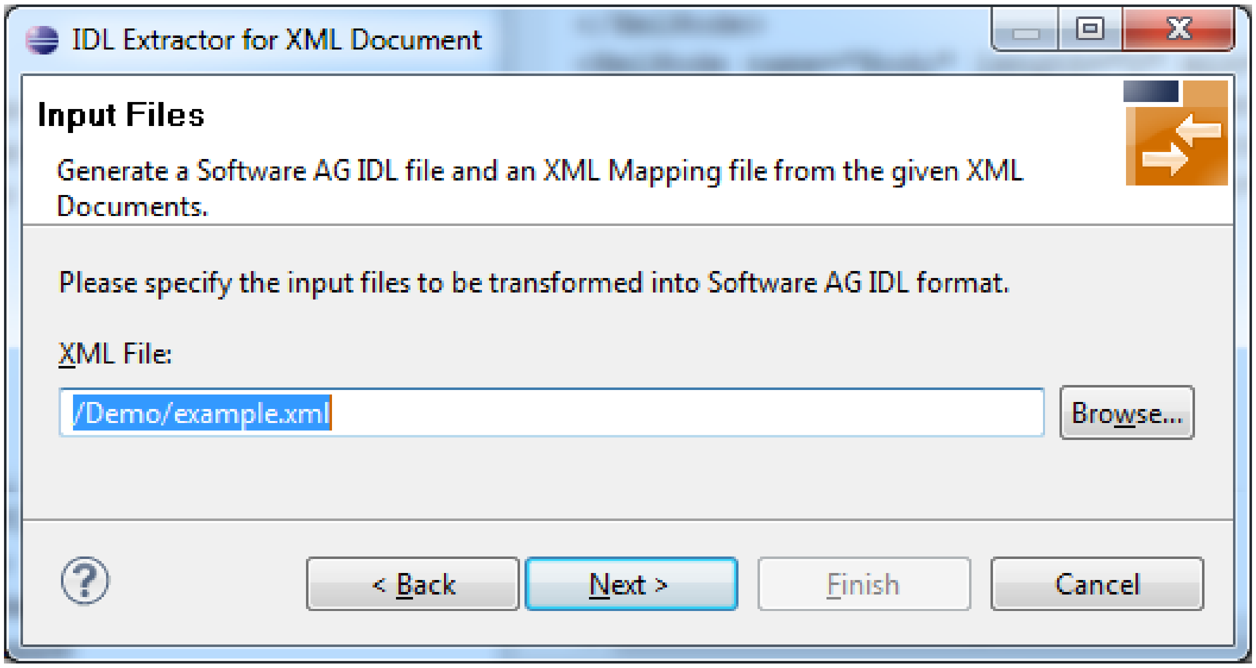
```
-Djavax.net.ssl.trustStore=<path to keystore>
-Djavax.net.ssl.trustStorePassword=<keystore password>
```

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 Document source file before you started the wizard, the file location is already present. Enter or browse for the XML Document source file. Continue with ***Step 4: Specify Output Files***.



Input Files

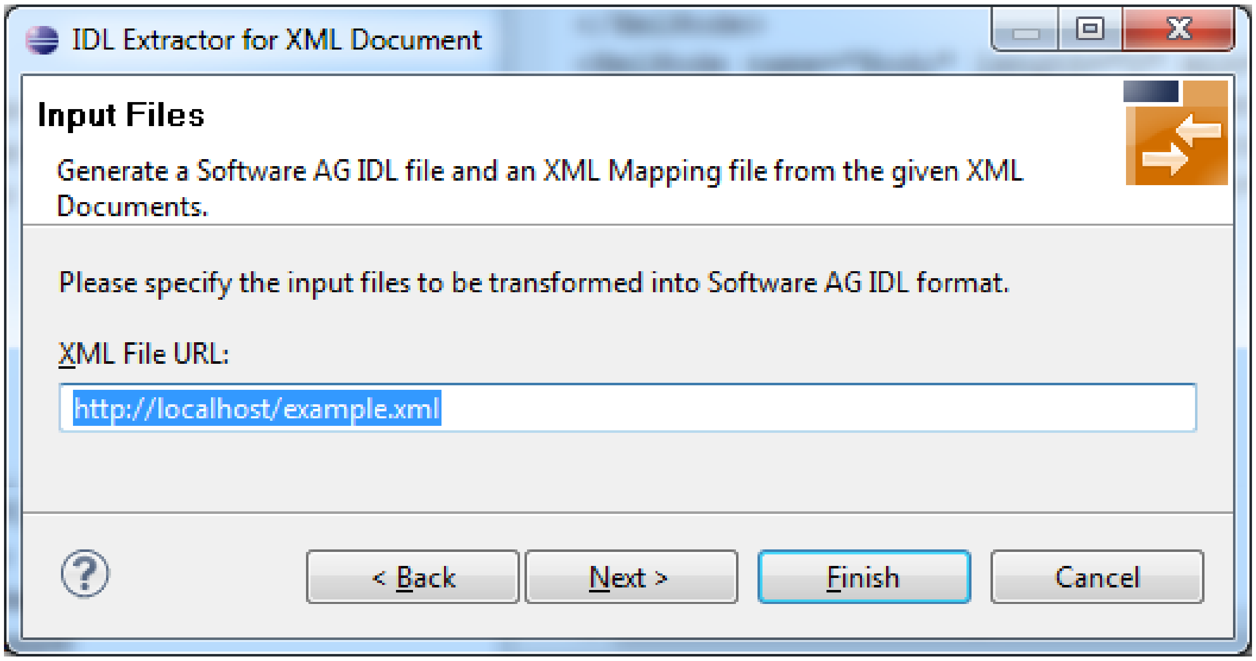
Generate a Software AG IDL file and an XML Mapping file from the given XML Documents.

Please specify the input files to be transformed into Software AG IDL format.

XML File: /Demo/example.xml

Step 3b: Specify XML File URL

Enter the URL for the XML Document source file.


IDL Extractor for XML Document

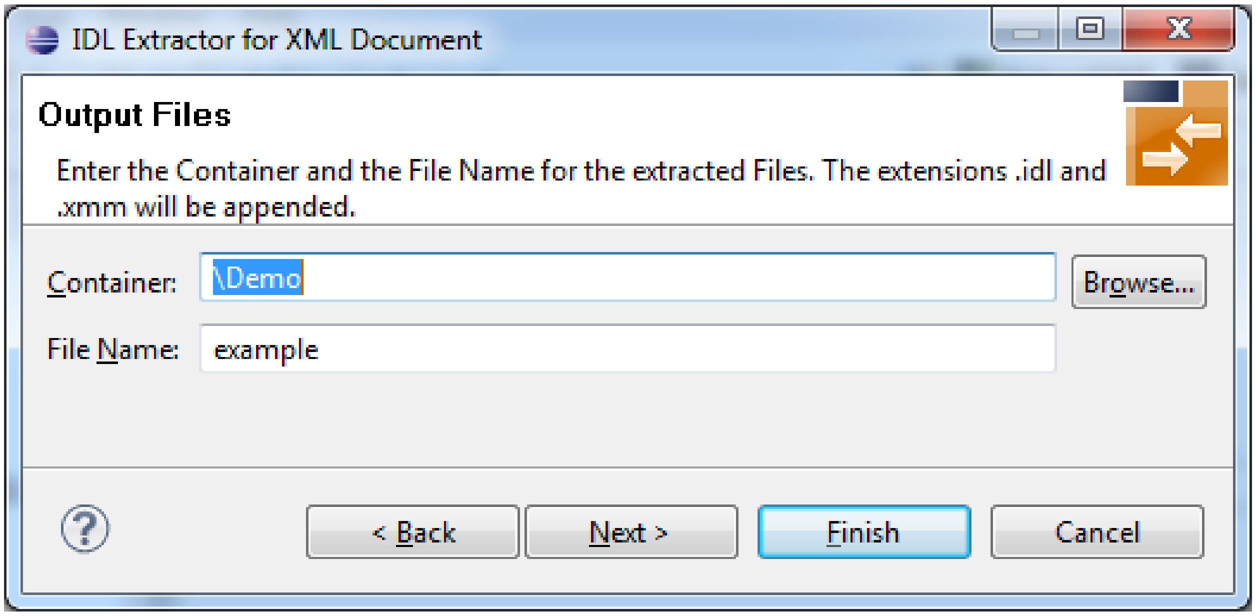
Input Files

Generate a Software AG IDL file and an XML Mapping file from the given XML Documents.

Please specify the input files to be transformed into Software AG IDL format.

XML File URL: http://localhost/example.xml

Step 4: Specify Output Files



Output Files

Enter the Container and the File Name for the extracted Files. The extensions .idl and .xmm will be appended.

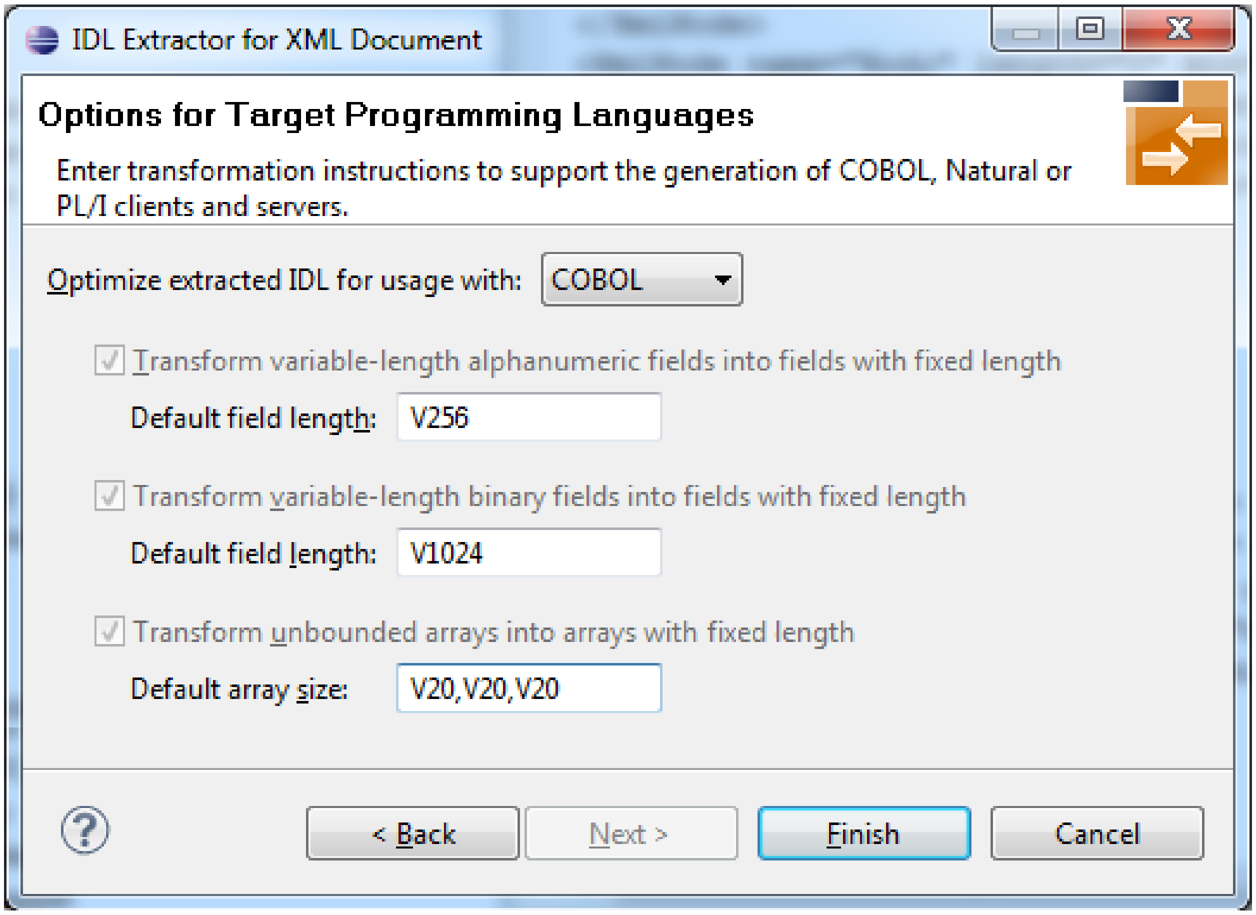
Container: \Demo

File Name: example

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.



Options for Target Programming Languages

Enter transformation instructions to support the generation of COBOL, Natural or PL/I clients and servers.

Optimize extracted IDL for usage with: COBOL

[✓] Transform variable-length alphanumeric fields into fields with fixed length

Default field length: V256

[✓] Transform variable-length binary fields into fields with fixed length

Default field length: V1024

[✓] Transform unbounded arrays into arrays with fixed length

Default array size: V20,V20,V20

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 Document 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 Document 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 Document.

Task	Command	Option	Description
Extract an IDL file and an XMM file from an XML Document.	`-extract:xml`	`-help`	Display this usage message.
		`-project`	Name of the project or subfolder where the IDL and XMM files are stored.

Example

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

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

The extracted IDL file and related XML Mapping (XMM) files will be stored in parallel to the XML document source file, e.g. in the project *Demo*.

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

4 XML Document to IDL Mapping

Extracting IDL from XML Document

The IDL Extractor for XML Document distinguishes between SOAP and pure XML. For XML documents, all parameters will be interpreted as strings and mapped according to specified transformation rules. See *Step 5: Specify Options for Target Programming Language*. SOAP documents, which contain `xsi:type` attributes, use the parameter mapping in the table *XML Schema Parameter Mapping* below. The IDL parameter directions are `IN-OUT` and the XML mapping directions are `IN` (Request) and `OUT` (Response), no Fault mapping trees will be created. If you try to import a SOAP Fault document, the document will be imported as a normal SOAP document.

XML Schema Parameter Mapping

WSDL / XML Schema	XMM	Software AG IDL
`binary, base64Binary`	`binary`	`BV` (or `BV`*n* or `B`*n*) (3)
`hexBinary` (1)	`binary`	`BV` (or `BV`*n* or `B`*n*) (3)
`boolean`	`boolean`	`L`
`date`	`date:yyyy-MM-dd` (2)	`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` (2)	`T`
`dateTime`	`dateTime:yyyy-MM-dd'T'HH:mm:ss` (2)	`T`
`gYearMonth`	`string`	`A8`
`gDay, gYear`	`string`	`A11`
`gMonth`	`string`	`A12`
`gMonthDay`	`string`	`A13`
`string` (and all types not listed here)	`string`	`AV` (or `AV`*n* or `A`*n*) (3)

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 document (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 the root tag of the XML document. If the document is SOAP dialect, the name of the first child element of the `<soapenv:Body>` `(xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/")` tag will be used.