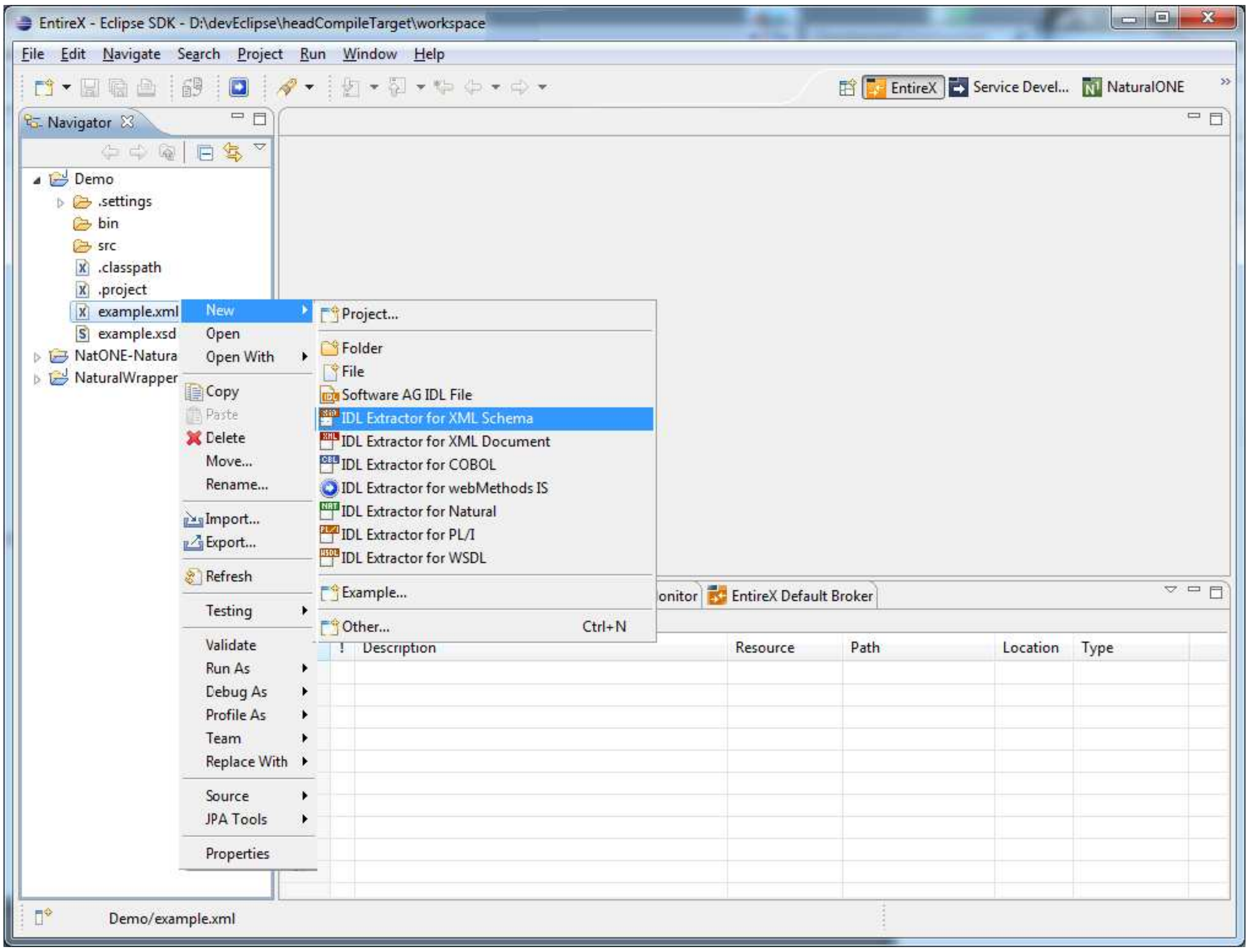
Using the Software AG IDL Extractor for XML Schema

Warning:
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.

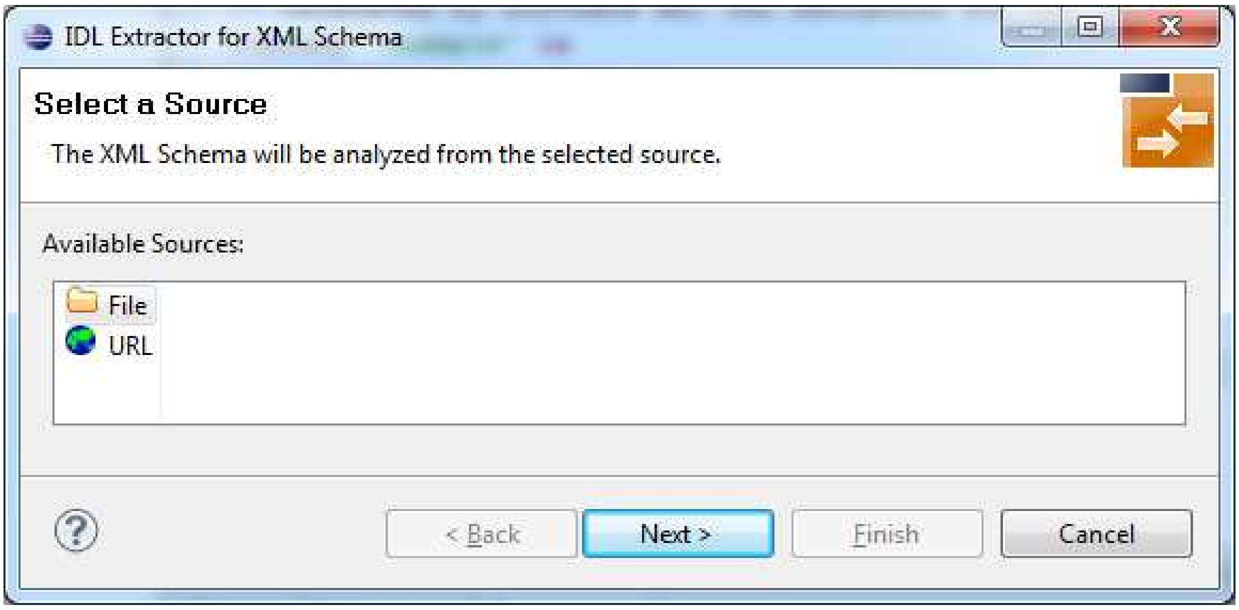
Step 1: Start the IDL Extractor for XML Schema

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**:


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

- **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>
     ```

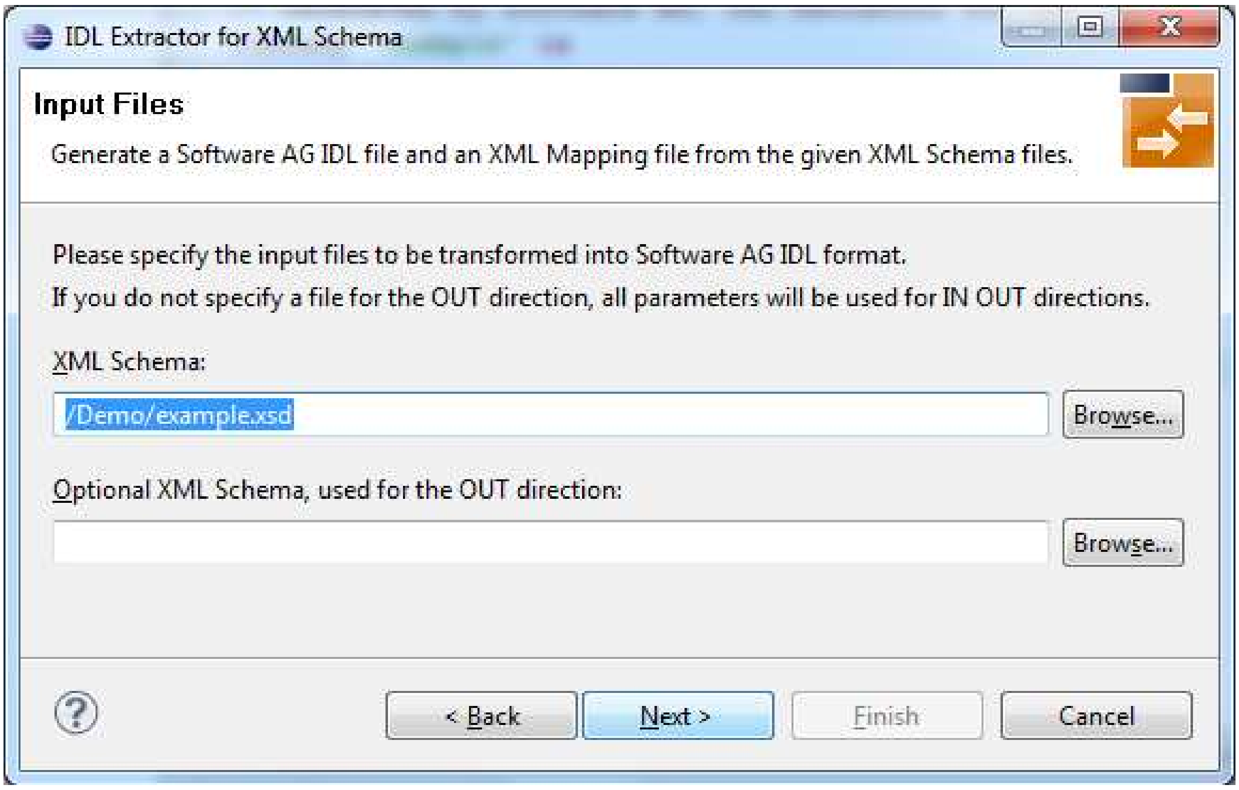
     If hostname verification for certification is to be disabled, also add the line:

     ```
     -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* contains all the parameters of the interface as IDL parameters (see `parameter-data-definition`) with its `direction` (see `attribute-list`).


IDL Extractor for XML Schema

Input Files

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

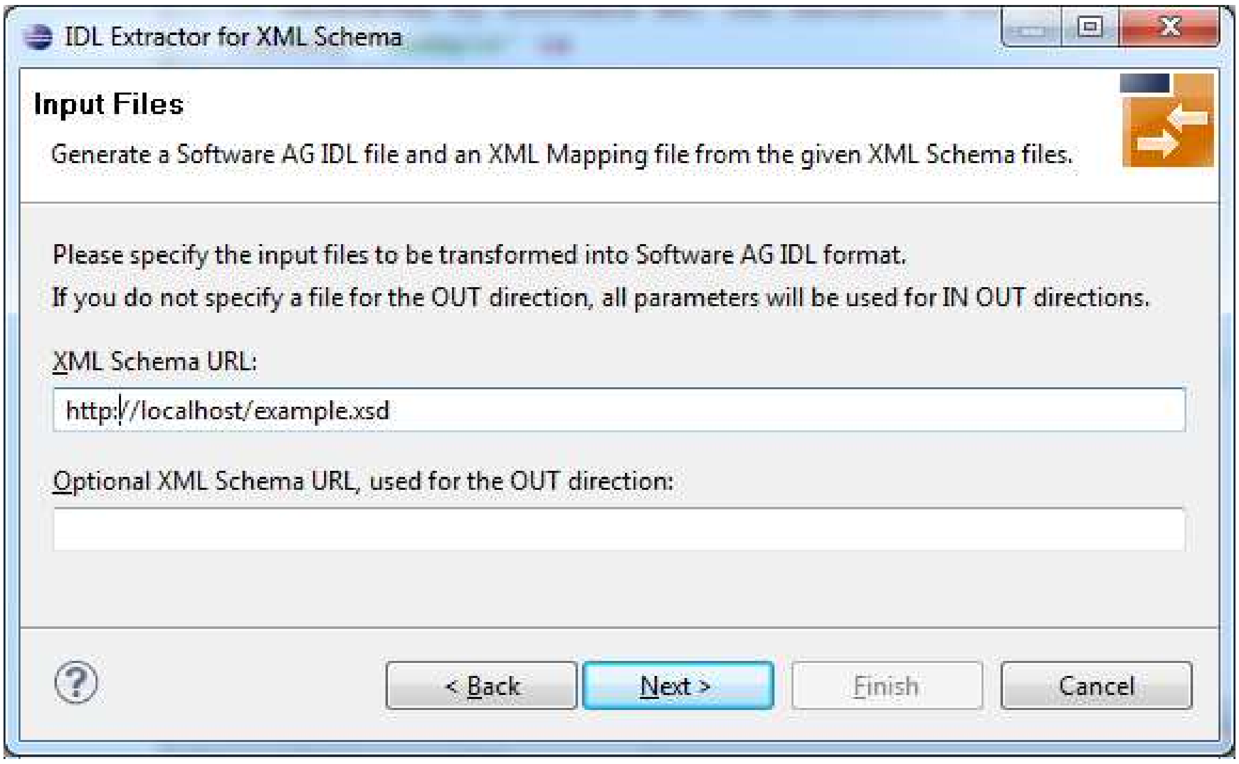
Please specify the input files to be transformed into Software AG IDL format.
If you do not specify a file for the OUT direction, all parameters will be used for IN OUT directions.

XML Schema: /Demo/example.xsd

Optional XML Schema, used for the OUT direction:

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.


IDL Extractor for XML Schema

Input Files

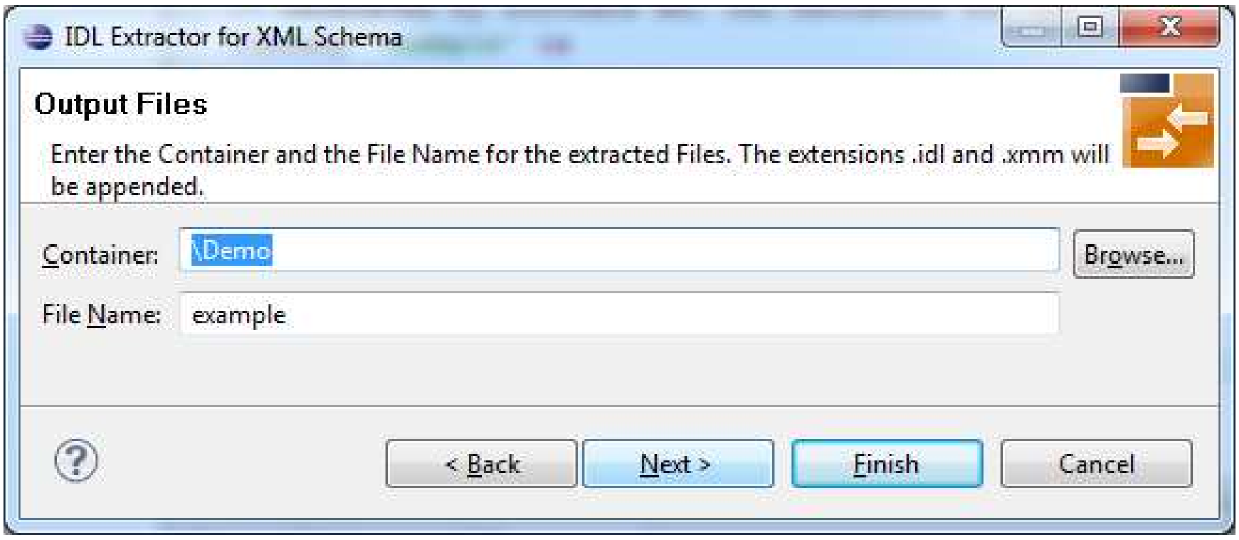
Generate a Software AG IDL file and an XML Mapping file from the given XML Schema files.

Please specify the input files to be transformed into Software AG IDL format.
If you do not specify a file for the OUT direction, all parameters will be used for IN OUT directions.

XML Schema URL: http://localhost/example.xsd

Optional XML Schema URL, used for the OUT direction:

Step 4: Specify Output Files


IDL Extractor for XML Schema

Output Files

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

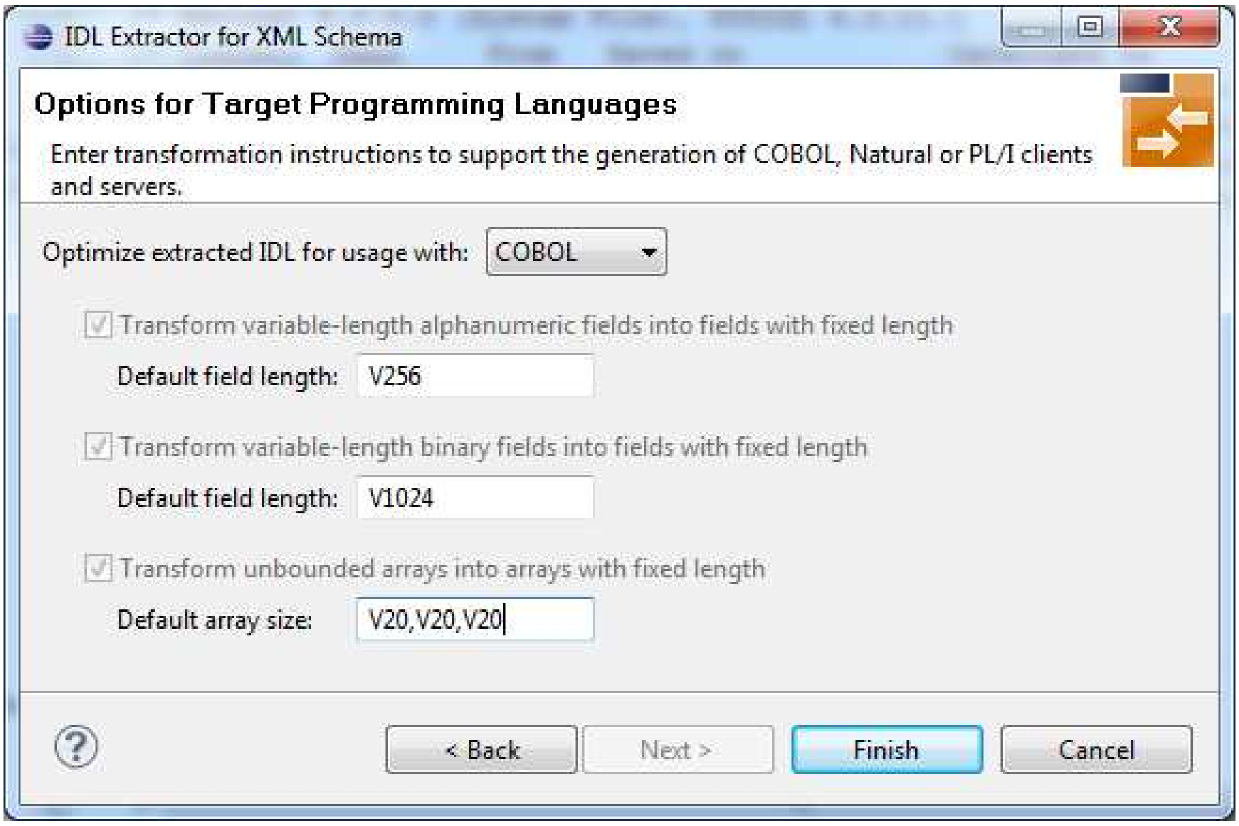
Container: \Demo

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.


IDL Extractor for XML Schema

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*

Warning:
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*. 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*, 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, clear 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* 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,* 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.