webMethods EntireX

Merging and Refactoring Software AG IDL

Version 10.5

October 2019

WEBMETHODS

This document applies to webMethods EntireX Version 10.5 and all subsequent releases.

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

Document ID: EXX-EEXXMERGEREFACTOR-105-20220422

Table of Contents

1 About this Documentation

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

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to "Developer Center", "User Center" or "Documentation".

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

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 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.

Refactoring a Single IDL File

In the case of a single IDL file, all programs and structures in every single library are checked if they contain identical groups. You will be asked for the name of the new IDL file.

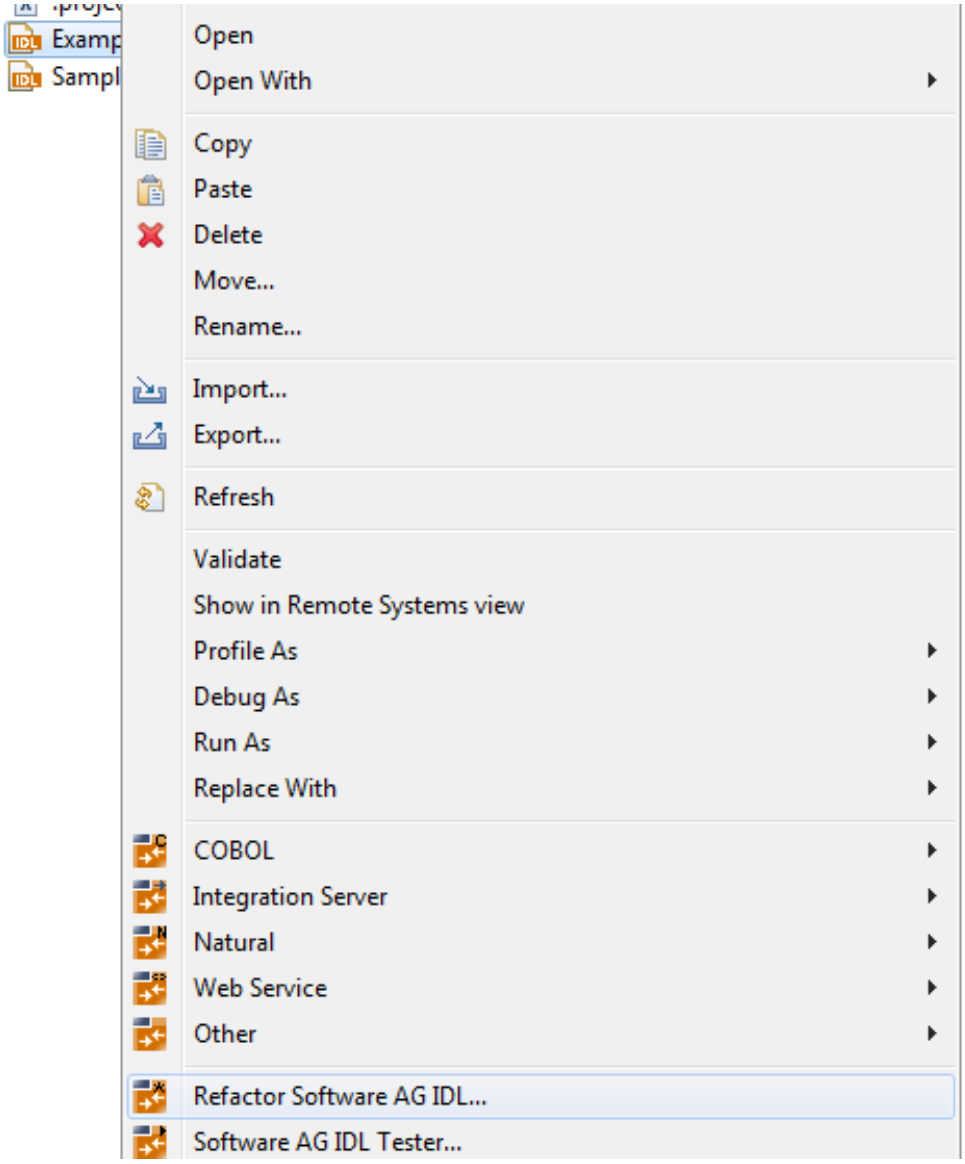
To illustrate refactoring, create an example IDL file *Example.idl*:

```
library 'EXAMPLE' is
  program 'SUM' is
    define data parameter
    1 Operands
      2 Operand1        (I4)  In
      2 Operand2        (I4)  In
    1 Function_Result   (I4)  Out
    end-define

  program 'SUBTRACTION' is
    define data parameter
    1 Ops
      2 Operand1        (I4)  In
      2 Operand2        (I4)  In
    1 Function_Result   (I4)  Out
    end-define

  program 'MULTIPLICATION' is
    define data parameter
    1 Operands
      2 Operand       (I4)  In
      2 Multiplier    (I4)  In
    1 Function_Result (I4)  Out
    end-define
```

By selecting this file in the Designer, the **Refactor Software AG IDL...** command in the context menu is enabled.



Executing the command and entering the target IDL file will result in a file *Example.refactored.idl*:

```
library 'EXAMPLE' is
  struct 'Operands' is
    define data parameter
    1 Operand1    (I4)
    1 Operand2    (I4)
    end-define

  program 'SUM' is
    define data parameter
    1 Operands          ('Operands')  In Out
    1 Function_Result   (I4)          Out
    end-define
```

```
  program 'SUBTRACTION' is
    define data parameter
    1 Ops               ('Operands')  In Out
    1 Function_Result   (I4)          Out
    end-define

  program 'MULTIPLICATION' is
    define data parameter
    1 Operands   In Out
      2 Operand       (I4)
      2 Multiplier    (I4)
    1 Function_Result (I4) Out
  end-define
```

As can be seen from above, the common group with parameters

```
Operand1    (I4)
Operand2    (I4)
```

is extracted as a single structure and the former groups are transformed to structure references. However, the group `Operands` from the `MULTIPLICATION` program is not replaced, because its members have different names, although parameters are equal in quantity and type.

Now, let us assume the *example.idl file* already contains a structure such as:

```
struct 'strct' is
   define data parameter
    1 Operand1    (I4)
    1 Operand2    (I4)
    end-define
```

Although its name differs, the structure's signature (number of parameters, parameter names and types) is the same as the one to be created. In this case, a new structure will not be created, but all references will point to the existing one - `strct`.

Merging and Refactoring Multiple IDL Files

If there is more than one IDL file, all files are first merged and then refactoring is run on the assembled file. You will be asked for the new IDL file's name.

To illustrate IDL merging, create two IDL files *Example1.idl* and *Example2.idl* with the following content:

- **Example1.idl**

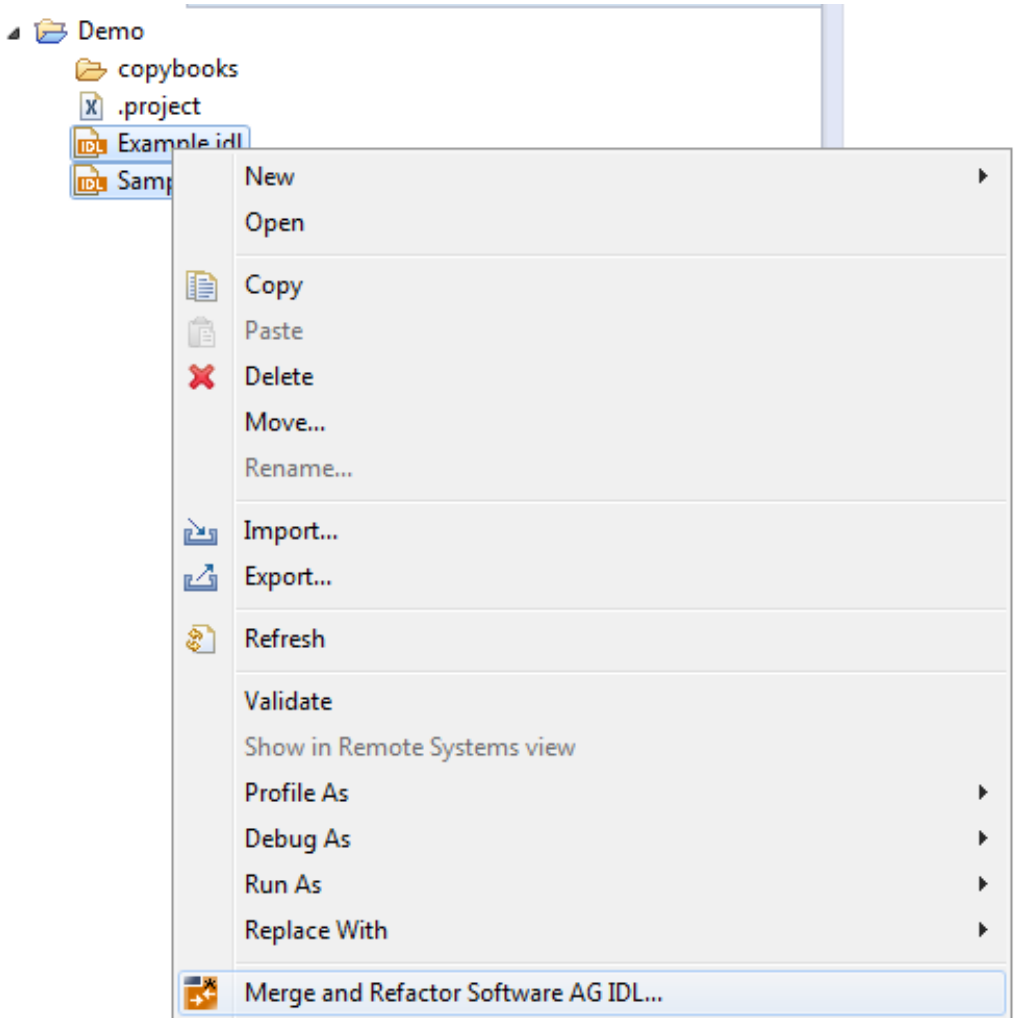
```
library 'EXAMPLE' is
    program 'SUM' is
        define data parameter
        1 Operands
          2 Operand1      (I4)  In
          2 Operand2      (I4)  In
        1 Function_Result (I4)  Out
        end-define

    program 'MULTIPLICATION' is
        define data parameter
        1 Operands
          2 Number        (I4)  In
          2 Multiplier    (I4)  In
        1 Function_Result (I4)  Out
        end-define
```

- **Example2.idl**

```
library 'EXAMPLE' is
    program 'SUBTRACTION' is
        define data parameter
        1 Ops
          2 Operand1      (I4)  In
          2 Operand2      (I4)  In
        1 Function_Result (I4)  Out
        end-define
```

Selecting these two files in the Designer will enable the **Merge and Refactor Software AG IDL...** command in the context menu.



Executing this command and entering the target IDL name will result in exactly the same file content as *Example.refactored.idl*. As the two source IDL files have libraries with one and the same name, an attempt to merge their programs is made. If successful, the result is a single `EXAMPLE` library containing all programs from the both libraries. This resulting library is then refactored as described above.

Notes on Merging

- Two libraries with different names will be simply copied into the target IDL file.
- For libraries with same names, an attempt to merge their programs and structures will be made. For a merge operation to be successful, the name of each program and structure must be unique. The only exception are pairs of programs or structures that are exactly the same. For example, if *Example2.idl* above contains the `MULTIPLICATION` program from *Example1.idl*, merging will be successfully completed.

Command-line Mode

See *Using EntireX in the Designer Command-line Mode* for the general command-line syntax. There are no specific command-line options for IDL Refactoring. If a single IDL file is passed, it will be refactored. If more than one IDL file is passed, they will be merged into one IDL file and it will be refactored. Example:

```
<workbench> -idl:refactor C:/Demo/example1.idl C:/Demo/example2.idl
```

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