Examples for Redesigning the Extracted Interfaces

This chapter gives examples on how to redesign the extracted interface. It covers the following topics:

- An Example for Set Constant
- An Example for Extracting Multiple Interfaces
- An Example for Extracting Natural REDEFINES

An Example for Set Constant

This example turns the CALC server into an ADD server. The CALC server used here can be found in *Basic RPC Server Examples* - *`CALC`, `SQUARE`* under *Client and Server Examples for Natural* in the Natural Wrapper documentation.

```
****************************************************************
*
*  CALC RPC Server Example for Natural
*
****************************************************************
DEFINE DATA
PARAMETER
  1 #OPERATION        (A1)
  1 #OPERAND1         (I4)
  1 #OPERAND2         (I4)
  1 #FUNCTION-RESULT  (I4)
LOCAL
  1 #TMP              (I4)
END-DEFINE
*
MOVE 0 TO #FUNCTION-RESULT
DECIDE ON FIRST VALUE OF #OPERATION
  VALUE '+'
    COMPUTE #FUNCTION-RESULT = #OPERAND1 + #OPERAND2
  VALUE '-'
    COMPUTE #FUNCTION-RESULT = #OPERAND1 - #OPERAND2
  VALUE '*'
    COMPUTE #FUNCTION-RESULT = #OPERAND1 * #OPERAND2
  VALUE '/'
    IF #OPERAND2 NE 0 THEN
      COMPUTE #FUNCTION-RESULT = #OPERAND1 / #OPERAND2
    END-IF
  VALUE '%'
    IF #OPERAND2 NE 0 THEN
      DIVIDE    #OPERAND2 INTO #OPERAND1 GIVING #TMP
      REMAINDER #FUNCTION-RESULT
    END-IF
  NONE VALUE
    IGNORE
END-DECIDE
END
```

Extraction without a redesign leads to an interface (IDL program) in which extracted parameters and Natural subprogram parameters are identical:

```
Library 'EXAMPLE' Is
   Program 'CALC' Is
      Define Data Parameter
         1 Operation        (A1) In
         1 Operand1         (I4) In
         1 Operand2         (I4) In
         1 Function_Result  (I4) Out
      End-Define
```

Calling such an interface with an RPC client means providing the ADD function as a value on the operation parameter in the same way legacy callers do.

After the redesign, the IDL looks as follows:

```
Library 'EXAMPLE' Is
   Program 'ADD' Is
      Define Data Parameter
         1 Operand1         (I4) In
         1 Operand2         (I4) In
         1 Function_Result  (I4) Out
      End-Define
```

This process is called "Redesigning the interface", which can be seen as manual step in the overall extraction process. See *Step 6: Redesign the Interface for Natural Subprograms (Optional)*.

Setting parameters to constant values and suppressing them in the IDL is often done together with multiple interfaces, where the function code or operation-code field has to be set to a constant. See *An Example for Extracting Multiple Interfaces* for a more advanced example.

To Redesign the interface and to set a constant for the parameter operation

1. Make sure to check **Redesign the interfaces** in *Step 5: Select Natural Subprograms from RPC Environment*.
2. In *Step 6: Redesign the Interface for Natural Subprograms (Optional)*, proceed as follows:

   Design interface for ADD Server

   - Choose **Set Constant** for parameter Operation and enter "+" as value.
   - Choose **Rename** from the toolbar and change the name to "ADD".

An Example for Extracting Multiple Interfaces

Multiple functions are often implemented in a single Natural subprogram, for example in *An Example for Set Constant*. The CALC server implements the functions ADD, SUBTRACT, MULTIPLY, DIVIDE and MODULO in one source.

Extraction without a redesign leads to an interface (IDL program) in which extracted parameters and Natural subprogram parameters are identical:

```
Library 'EXAMPLE' Is
   Program 'CALC' Is
      Define Data Parameter
         1 Operation        (A1) In
         1 Operand1         (I4) In
         1 Operand2         (I4) In
         1 Function_Result  (I4) Out
      End-Define
```

Calling such an interface with an RPC client means providing the function (ADD, SUBTRACT, MULTIPLY, DIVIDE or MODULO) as a value on the operation parameter in the same way legacy callers do.

The interface above would

- result in a class with a single method if it was wrapped with an object-oriented wrapper (*Java Wrapper*, *.NET Wrapper*, *DCOM Wrapper*).
- offer CALC as only operation if it was used as web service.

A modern design of the same interface would provide more specialized interfaces. Each interface may have fewer parameters compared to the original Natural subprogram. Parameters that are obsolete for a function are not offered in the corresponding interface, for example the parameter `Operation` in our example. This keeps the interface lean and clear and makes it easier to use from the perspective of an RPC client.

This process is called "Redesigning the interface", which can be seen as manual step in the overall extraction process. See *Step 6: Redesign the Interface for Natural Subprograms (Optional)*.

After the redesign, the IDL looks as follows:

```
Library 'EXAMPLE' Is
   Program 'ADD' Is
      Define Data Parameter
         1 Operand1         (I4) In
         1 Operand2         (I4) In
         1 Function_Result  (I4) Out
      End-Define
   Program 'SUBTRACT'
      Define Data Parameter
         1 Operand1         (I4) In
         1 Operand2         (I4) In
         1 Function_Result  (I4) Out
      End-Define
   Program 'MULTIPLY'
      Define Data Parameter
         1 Operand1         (I4) In
         1 Operand2         (I4) In
         1 Function_Result  (I4) Out
      End-Define
   Program 'DIVIDE'
      Define Data Parameter
         1 Operand1         (I4) In
         1 Operand2         (I4) In
         1 Function_Result  (I4) Out
      End-Define
   Program 'MODULO'
      Define Data Parameter
      1 Operand1         (I4) In
      1 Operand2         (I4) In
      1 Function_Result  (I4) Out
   End-Define
```

By redesigning the interface you can turn the legacy interface of the CALC server into a more modern, object-oriented interface. In case the redesigned interface above is wrapped

- with an object-oriented wrapper (*Java Wrapper*, *.NET Wrapper*, *DCOM Wrapper*), this results in a class with five methods – `ADD`, `SUBTRACT`, `MULTIPLY`, `DIVDE` and `MODULO`, a real class/method design.
- as web service, also five operations are offered.

To Redesign the interface and extract multiple interfaces

1. Make sure to check **Redesign the interfaces** in *Step 5: Select Natural Subprograms from RPC Environment* in the extractor wizard (see *Extracting Software AG IDL File from an Existing Natural RPC Environment*).
2. In *Step 6: Redesign the Interface for Natural Subprograms (Optional)*, proceed as follows:
   - **Design interface for ADD**
     - Choose **Set Constant** for parameter **Operation** and enter ’+’ as value.
     - Choose **Rename** from the toolbar and change the name to `ADD`
   - **Design interface for SUBTRACT**
     - Choose **Set Constant** for parameter **Operation** and enter ’-’ as value.
     - Choose **Rename** from the toolbar and change the name to SUBTRACT
   - **To design MULTIPLY, DIVIDE or MODULO, proceed as for SUBTRACT**
     - Enter ’*’ as value for **Operation** and rename it to `MULTIPLY`.
     - Enter ’/’ as value for **Operation** and rename it to `DIVIDE`.
     - Enter ’%’ as value for **Operation** and rename it to `MODULO`.

An Example for Extracting Natural REDEFINES

`REDEFINE`s are often used in Natural. The example below illustrates a simple redefinition where the parameter `#BASE-FIELD` is redefined by the fields `FILLER-1` thru `R-P3-01`.

```
DEFINE DATA PARAMETER
1 #BASE-FIELD            (A161) /* first parameter
1 REDEFINE #BASE-FIELD            /* second parameter
  2 FILLER-1             (A4)
  2 FILLER-2             (A60)
  2 R-P1-01              (A1)
  2 R-P2-01              (A10)
  2 R-P3-01              (I4)
END-DEFINE
```

You can select a single `REDEFINE` path of the same base field for one interface only. If you require more than one `REDEFINE` path of the same base field, extract multiple interfaces – for each `REDEFINE` path a separate interface, see *An Example for Extracting Multiple Interfaces*.

To redesign the interface and extract a `REDEFINE` path

- Choose **Map to In**, **Map to Out** or **Map to InOut** either for
  - the `REDEFINE` base parameter, here parameter `#BASE-FIELD (A161) /* first parameter`, or
  - any `REDEFINE` path, here `REDEFINE#BASE-FIELD /* second parameter`