webMethods EntireX

Using EntireX RPC for RPG under IBM i

Version 10.7

October 2020

WEBMETHODS

This document applies to webMethods EntireX Version 10.7 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-RPC-107-20220422RPG

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 Using EntireX RPC for RPG under IBM i

The EntireX RPC Server for IBM i of EntireX version 7.1.1 running under IBM i (AS/400) is no longer supported. We strongly recommend using the new *RPC Server for AS/400* or the EntireX Adapter. See also *Connection Parameters for AS/400 Connections*.

Creating a Sample Server in RPG

This section describes how to build a server application using the IBM i ILE language RPG. The server will be named `CALC_RPG`. Its functionality and implementation is based on the ILE COBOL server CALC as described in the section *Using the COBOL Wrapper*.

The function `CALC_RPG` is a calculator that can add, subtract, multiply and divide two binary values `PIC S9(8) BINARY` and return a result.

This section tells you how to

- Create the Client/Server Interface
- Create the Server
- Compile and Link the Server

Create the Client/Server Interface

The PLIST in the RPG program is a good source of type information for the interface when you create the Software AG IDL file:

```
...
 D OPERATOR        S              1A
 D OPERAND_1       S             10I 0
 D OPERAND_2       S             10I 0
 D FCT_RESULT      S             10I 0
  *
 C     *ENTRY        PLIST
 C                   PARM                    OPERATOR
 C                   PARM                    OPERAND_1
 C                   PARM                    OPERAND_2
 C                   PARM                    FCT_RESULT
....
```

The assumption is made that the program is implemented in library `EXAMPLE`. Convert the linkage section function above to Software AG IDL syntax as follows:

```
Library 'EXAMPLE' Is
 Program 'CALC_RPG' Is
  Define Data Parameter
   1 Operator         (A1) In
   1 Operand_1        (I4) In
   1 Operand_2        (I4) In
   1 Function_Result  (I4) Out
  End-Define
```

Note: A 10-digit RPG integer takes 4 bytes, so it must be mapped to an (I4) IDL field definition.

For details on how IDL field definitions are mapped to RPG elementary field items, see *Software AG IDL to RPG Mapping*.

Create the Server

The server is implemented as an ILE RPG program of type *PGM.

For our IDL example CALC, the implemented server looks similar to the example below.

```
 *----------------------------------------------------
 * Member          CALC_RPG
 * Description     Calculation Engine.
 *
 * Author          (c) Software AG
 * Platform        OS/400
 *
 * UUU YYYY-MM-DD  History
 * HBA 2003-05-20  Created
 *
 *--------------- CALC Interface ------------------------------
D OPERATOR        S              1A
D OPERAND_1       S             10I 0
D OPERAND_2       S             10I 0
D FCT_RESULT      S             10I 0
 *-------------------------------------------------------------
C     *ENTRY        PLIST
C                   PARM                    OPERATOR
C                   PARM                    OPERAND_1
C                   PARM                    OPERAND_2
C                   PARM                    FCT_RESULT
C                   CLEAR                   FCT_RESULT
 *
C                   SELECT
C                   WHEN      OPERATOR = '+'
C                   EVAL      FCT_RESULT = OPERAND_1 + OPERAND_2
C                   WHEN      OPERATOR = '-'
C                   EVAL      FCT_RESULT = OPERAND_1 - OPERAND_2
C                   WHEN      OPERATOR = '*'
```

```
C                   EVAL      FCT_RESULT = OPERAND_1 * OPERAND_2
C                   WHEN      OPERATOR = '/'
C                   IF        OPERAND_2 <> *ZERO
C                   EVAL      FCT_RESULT = OPERAND_1 / OPERAND_2
C                   ENDIF
C                   ENDSL
 *
C     PGM_EX        TAG
C                   MOVE      *ON           *INLR
```

Compile and Link the Server

Compile the server source using the IBM i command `CRTRPGMOD` (create bound RPG module) and bind it as a dynamically callable program of type `*PGM` using the command `CRTPGM`. Make sure your server fulfills all requirements to be callable by the IBM i host server.

As an alternative to the commands `CRTRPGMOD` and `CRTPGM`, you can use the command `CRTBNDRPG` to compile and bind RPG sources in one step.

Name the resulting server program like the program name in the IDL file and put it in a library whose name corresponds to the library name in the IDL file.

Example

If a client performs an RPC which is based on the IDL program `CALC_RPG` in the IDL library `EXAMPLE`, the *RPC Server for AS/400* or the EntireX Adapter (see *Connection Parameters for AS/400 Connections*) will dynamically try to execute the ILE program `CALC_RPG` in the IBM i library `EXAMPLE`. If no corresponding program can be found, the access fails.

Verifying the Server

To verify the server program `CALC_RPG`, use the *EntireX IDL Tester* when the RPC Server for AS/400 and EntireX Broker are used.

To verify the server

1. Confirm that an EntireX Broker and an *RPC Server for AS/400* are available in your network.
2. Make sure the IBM i host is started on your IBM i machine.
3. Create a *Software AG IDL File* in the IDL Editor documentation using the *IDL Editor* as described under ***Create the Client/Server Interface***.
4. Start the IDL Tester using the IDL file created in step 3.

Note: When using the EntireX Adapter, use the **Service Development** perspective of the Software AG Designer for testing.

Software AG IDL to RPG Mapping

This section describes the specific mapping of Software AG IDL data types, groups, arrays and structures to the RPG programming language. See also hints and restrictions on the Software AG IDL data types valid for all programming language bindings under *IDL Data Types* in the IDL Editor documentation..

Mapping IDL Data Types to RPG Data Types

In the table below, the following metasymbols and informal terms are used for the IDL.

- The metasymbols "[" and "]" enclose optional lexical entities.
- The informal term *number* (or in some cases *number1*.*number2*) is a sequence of numeric characters, for example 123.

Software AG IDL	Description	RPG Data Type	See Notes
A*number*	Alphanumeric	*number*A	
AV	Alphanumeric variable length	not supported	
AV[*number*]	Alphanumeric variable length with maximum length	*number*A	
B*number*	Binary	*number*B	
BV	Binary variable length	not supported	
BV[*number*]	Binary variable length with maximum length	*number*B	
D	Date	8U	1
F4	Floating point (small)	not supported	
F8	Floating point (large)	not supported	
I1	Integer (small)	3I 0	
I2	Integer (medium)	5I 0	
I4	Integer (large)	10I 0	

Software AG IDL	Description	RPG Data Type	See Notes
K*number*	Kanji	*number*A	
KV	Kanji variable length	not supported	
KV[*number*]	Kanji variable length with maximum length	*number*A	
L	Logical	not supported	
N*number1*[.*number2*]	Unpacked decimal	*number1*S *number2*	3
NU*number1*[.*number2*]	Unpacked decimal unsigned	*number1*S *number2*	3
P*number1*[.*number2*]	Packed decimal	*number1*P *number2*	3
PU*number1*[.*number2*]	Packed decimal unsigned	*number1*P *number2*	3
T	Time	15U	2

See also hints and restrictions on the Software AG IDL data types valid for all programming language bindings under *IDL Data Types* in the IDL Editor documentation.

Notes:

1. The Date corresponds to the format 8U (unpacked decimal unsigned). The value contained has the form `YYYYMMDD`.
2. The Time corresponds to the format `15U` (unpacked decimal unsigned). The value contained has the form `YYYYMMDDHHIISST`.
3. For RPG, the total number of digits (`number1+number2`) is 18. This is lower than the maximum of 99 supported by EntireX. See *IDL Data Types*.

   If you connect two endpoints, the total number of digits used must be lower or equal than the maxima of both endpoints. For the supported total number of digits for endpoints, see the notes under data types N, NU, P and PU in section *Mapping IDL Data Types* in the respective Wrapper or language-specific documentation.

Mapping Program and Library Names

Do not use the special characters '#', '$', '&', '+', '-', '.', '/' and '@' within names of programs and libraries in the IDL file. These characters are not allowed within names of server programs and libraries created on IBM i.

Mapping Arrays, Groups and Structures

- Fixed arrays within the Software AG IDL file are mapped to fixed RPG tables. See the `array-definition` under *Software AG IDL Grammar* in the IDL Editor documentation for the syntax on how to describe fixed arrays within the Software AG IDL file and refer to fixed-bound-array-index.
- Unbounded arrays without a maximum are not supported.

Mapping Arrays, Groups and Structures

Groups within the Software AG IDL file are mapped to RPG tables. See the `group-parameter-definition` under *Software AG IDL Grammar* in the IDL Editor documentation for the syntax on how to describe groups within the Software AG IDL file.

Example

The following IDL definition shows a simple group structure:

```
Library 'EXAMPLE' Is
 Program 'GROUP' Is
   Define Data Parameter
    1 MYGROUP
      2 PART1        (A10) In Out
      2 PART2        (A10) In Out
   End-Define
```

The following source file excerpt from a sample RPG program named `GROUP` shows the corresponding field definitions and the entry parameter list:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+
 *
D MYGROUP         DS
D PART1                          10
D PART2                          10
 *
 ***********************************************
 *
C     *ENTRY        PLIST
C                   PARM                    MYGROUP
 *
```

Mapping Arrays, Groups and Structures

Structures within the Software AG IDL file are mapped to RPG tables like groups. See the structure definition for the syntax on how to describe structures within the Software AG IDL file.

Mapping the Direction Attributes In, Out, InOut

The IDL syntax allows you to define parameters as `IN` parameters, `OUT` parameters, or `IN OUT` parameters (which is the default if nothing is specified). This direction specification is reflected in the stubless call of the RPC Server as follows:

- Direction attributes do not change the call interface because parameters are always treated as "called by reference".
- Usage of direction attributes may be useful to reduce data traffic between RPC client and RPC server.
- Parameters with the `IN` attribute are sent from the RPC client to the RPC server.
- Parameters with the `OUT` attribute are sent from the RPC server to the RPC client.
- Parameters with the `IN` and `OUT` attribute are sent from the RPC client to the RPC server and then back to the RPC client.

Note that only the direction information of the top-level fields (Level 1) is relevant. Group fields always inherit the specification from their parent. A different specification is ignored.

See the `attribute-list` under *Software AG IDL Grammar* in the IDL Editor documentation for the syntax on how to describe attributes within the Software AG IDL file and refer to `direction-attribute`.