Using EntireX RPC for RPG under IBM i

This chapter covers the following topics:

- EntireX RPC Server Example
- Creating a Sample Server in RPG
- Verifying the Server
- Software AG IDL to RPG Mapping

See also *Administering the EntireX RPC Server*.

EntireX RPC Server Example

An EntireX RPC server example for RPG is provided in library EXAMPLE of the Developer's Kit for IBM i.

The source file *QCBLLESRC* contains the server-side implementation of the procedure CALC. The associated IDL definition is the same as for the client side.

For details on the IBM i installation kit, see *Step 1: Restore the EXAMPLE Library*.

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 *Using EntireX RPC for RPG under IBM i*.

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. (It is contained in the member CALC_RPG in the source file EXAMPLE/QRPGLESRC).

```
  *---------------------------------------------------
  * 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
```

The servers are running in a multithreaded environment. Therefore your application server programs must be thread-safe. This implies that all commands and subprograms accessed in your servers must allow multithreads.

For RPG multithreading issues in RPG, see the IBM documentation *Multithreaded Applications* (V4R4 in this case).

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. See the example procedure `BIND_RCALC` under *Step 1: Restore the EXAMPLE Library*.

> **Important:**
> When you are linking/binding servers, the CRTPGM parameter ACTGRP(*CALLER) must be specified. This guarantees that the server application runs in the same activation group as the calling RPC 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.

If you put the server program in a library other than the IDL library, you can reroute the call using the server parameter `Library=FIX( MyLib)`. In this case, the library name sent with the client request is ignored.

Example:

If a client performs an RPC which is based on the IDL program CALC_RPG in the IDL library EXAMPLE, the remote RPC server 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.

The principles of calling a server are described under *Administering the EntireX RPC Server*.

See *Step 3: Verify the RPC Server using COBOL* for how to start an RPC server that can execute the server program CALC_RPG.

Verifying the Server

To verify the server program CALC_RPG, Software AG recommends that you use a client Java program from the EntireX Workbench. See *EntireX Workbench*.

To start the Java Client

1. Confirm that an EntireX Broker and an EntireX RPC server are available in your network.
2. Start/submit an RPC server on your IBM i machine as described under *Starting the RPC Server under IBM i*.
3. Open the EntireX Workbench and find the sample IDL file *example.idl* delivered with the Windows installation kit.

   Copy the IDL program CALC to CALC_RPG.
4. From the menu bar choose **Java** and **Generate and run Test**. After compilation, a pop-up menu will offer you the option of running the Java client program. Select the server program CALC_RPG.
5. On the Java calculation menu, specify the numbers you want to compute and choose CALL. The RPC request will be sent to the RPC Server on your IBM i, which will run the ILE server program CALC_RPG in library EXAMPLE. If the Broker connection fails, you will receive an appropriate error message.

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 the remarks and hints on the Software AG IDL data types valid for all language bindings found in *Software AG IDL File*.

The following topics are covered here:

- Mapping IDL Data Types to RPG Data Types
- Mapping Program and Library Names
- Mapping Arrays, Groups and Structures
- Mapping Arrays, Groups and Structures
- Mapping Arrays, Groups and Structures
- Mapping the Direction Attributes In, Out, InOut

Mapping IDL Data Types to RPG Data Types

In the table below, the following metasymbols and informal terms are used for the Software AG 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	2
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
		8P	10
		21U	11
F4	Floating point (small)	4F	4,5
F8	Floating point (large)	8F	4,5
I1	Integer (small)	3I 0	
I2	Integer (medium)	5I 0	6
I4	Integer (large)	10I 0	6
K*number*	Kanji	*number*A	
KV	Kanji variable length	not supported	
KV[*number*]	Kanji variable length with maximum length	*number*A	2
L	Logical	not supported	
N*number1*[.*number2*]	Unpacked decimal	*number1*S *number2*	7,8
NU*number1*[.*number2*]	Unpacked decimal unsigned	*number1*U *number2*	7
P*number1*[.*number2*]	Packed decimal	*number1*P *number2*	7,9
PU*number1*[.*number2*]	Packed decimal unsigned	*number1*P *number2*	7
T	Time	15U	3
		15P	12
		21U	13

See also the hints and restrictions under *IDL Data Types* valid for all language bindings.

Notes:

1. For clients the Date corresponds to the format 8U (unpacked decimal unsigned). The value contained has the form `YYYYMMDD`.
2. To use variable length, specify the keyword `VARYING` in your RPG field definition.
3. For clients the Time corresponds to the format `15U` (unpacked decimal unsigned). The value contained has the form `YYYYMMDDHHIISST`.
4. When floating-point data types are used, rounding errors can occur, so that the values of senders and receivers might differ slightly.
5. For servers the alignment for floating-point variables can be customized. See the `F4-ALIGNED` and `F8-ALIGNED` runoption parameter of your EntireX RPC Server under CICS, Batch, IBM i.
6. For servers the alignment for integer variables can be customized. See the `I2-ALIGNED` and `I4-ALIGNED` runoption parameter of your EntireX RPC Server under CICS, Batch, IBM i.
7. 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* to target language environment C | CL | COBOL | DCOM | .NET | Java | Natural | PL/I | RPG | XML.

8. For servers the mapping for unpacked-decimal variables depends on the setting of the `N-SIGNED` runoption parameter of your EntireX RPC Server under CICS, Batch, IBM i.
9. For servers the mapping for packed-decimal variables depends on the setting of the `P-SIGNED` runoption parameter of your EntireX RPC Server under CICS, Batch, IBM i.
10. For servers the date can be mapped to the format `8P` (packed-decimal) depending on the setting of the `COBOL-TIME` runoption parameter of your EntireX RPC Server under CICS, Batch, IBM i. The value contained is the number of dates since start of the Gregorian calendar 1. Jan.1582. See also *Software AG IDL File*.
11. For servers the date can be mapped to the format 21U (unpacked decimal unsigned) depending on the setting of the `COBOL-TIME` runoption parameter of your EntireX RPC Server under CICS, Batch, IBM i. The value contained is a date in the form `YYYYMMDD0000000000000`.
12. For servers the time can be mapped to the format 15P (packed-decimal) depending on the setting of the `COBOL-TIME` runoption parameter of your EntireX RPC Server under CICS, Batch, IBM i. The value contained is the count of tenth of seconds starting from 2.1.0000 0:00:00.0. See also *Software AG IDL File*.
13. For servers the date/time can be mapped to the format `21U` (unpacked decimal unsigned) depending on the setting of the `COBOL-TIME` runoption parameter of your EntireX RPC Server under CICS, Batch, IBM i. The value contained is a date in the form `YYYYMMDDhhmmsst00000`.

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* 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* for the syntax on how to describe attributes within the Software AG IDL file and refer to `direction-attribute`.