RPG Samples for Rendezvous

The Rendezvous installation package for IBM i includes ILE RPG samples (source files and executables).

The samples `tibrvlisten.rpgle` and `tibrvsend.rpgle` parallel the functionality in `tibrvlisten.c` and `tibrvsend.c`.

The file `TIBRV.rpgle` contains API prototypes for the calls in these samples.

Topics

Listen Program

The program `tibrvlisten` receives Rendezvous messages and outputs them.

Copies of `tibrvlisten.rpgle` exist both in `QRPGLESRC(TIBRVLISTE)` and in the IFS (at *RV_IFS_HOME*`/src/examples/rpg`).

This implementation of `tibrvlisten` illustrates an RPG-style call interface with positional parameters.

For simplicity, the program uses the `DSPLY` operation to the `QSYSOPR` message queue for all output. To monitor this information without explicitly displaying the message queue, consider setting the queue to `*BREAK` mode:

```
CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(0)
```

Syntax

`call tibrvlisten` *service network daemon subject1 subject_list*

Parameters

The first four parameters all accept either a specifc value, or the keyword `*DEFAULT`.

Parameter	Description
service	Specify the service number. For details, see Service Parameter in TIBCO Rendezvous Concepts. The program interprets the keyword `*DEFAULT` as 7500.
network	Specify the network name. For details, see Network Parameter in TIBCO Rendezvous Concepts. The program interprets the keyword `*DEFAULT` as the null string.
daemon	Specify the daemon parameter. For details, see Daemon Parameter in TIBCO Rendezvous Concepts. The program interprets the keyword `*DEFAULT` as `localhost:7500`.
subject1	The program listens for messages with this subject name. The program interprets the keyword `*DEFAULT` as `'your.subject'`.
subject_list	Optional. When present, the program also listens to these subject names. You may supply between zero and four additional subjects.

Call Examples

The first three examples specify equivalent behavior—the program uses the default values for all parameters.

```
ADDLIBLE tibrv_lib
CALL PGM(TIBRVLISTE)

ADDLIBLE tibrv_lib
CALL PGM(TIBRVLISTE) PARM(*DEFAULT *default *DEFAULT *DEFAULT)

ADDLIBLE tibrv_lib
CALL PGM(TIBRVLISTE) PARM(*DEFAULT *DEFAULT *DEFAULT )
```

In the next example, the user supplies a non-default network parameter, and a second listen subject.

```
ADDLIBLE tibrv_lib
CALL PGM(TIBRVLISTE) PARM('7500' *DEFAULT
'myhost.my_enterprise.com:7500' 'our.subject' 'your.subject')
```

Send Program

The send program sends Rendezvous messages.

Copies of `tibrvsend.rpgle` exist both in `QRPGLESRC(TIBRVSEND)` and in the IFS (at *RV_IFS_HOMERV_IFS_HOME*`/src/examples/rpg`).

In contrast to the listen program, this implementation illustrates a C-style call interface with parameter switches. The code uses fixed-column calculation specifications.

For simplicity, the program uses the `DSPLY` operation to the `*EXT` message queue to communicate with the user.

Syntax

```
call  tibrvsend -service service -network network -daemon daemon
                subject
                msg_list
```

Parameters

The first three parameters (`-service`, `-network` and `-daemon`) are switched parameters; pair each switch with its value. You may enter these three pairs in any order. If you omit any of these three parameters, the program supplies default values.

Parameter	Description
`-service` *service*	Optional. Specify the service number. For details, see Service Parameter in TIBCO Rendezvous Concepts. When absent, the default value is 7500.
`-network` *network*	Optional. Specify the network name. For details, see Network Parameter in TIBCO Rendezvous Concepts. When absent, the default value is the null string.
`-daemon` *daemon*	Optional. Specify the daemon parameter. For details, see Daemon Parameter in TIBCO Rendezvous Concepts. When absent, the default value is `localhost:7500`.

Parameter	Description
subject	Required. The program interprets the first non-switched parameter as the send subject name.
msg_list	Required. The program interprets all subsequent parameters as messages. It sends these messages on the send subject. The program parses the message strings, and supplies field names. You must supply the characters `**` to delimit each message string. Embedded space characters are permitted.

Call Example

Notice that this example omits the `-service` and `-network` parameters. It supplies two messages.

```
ADDLIBLE tibrv_lib

CALL PGM(TIBRVSEND) PARM('-daemon' 'myhost.my_enterprise.com:7500'
'b.c.d' 'message 1**' 'message 2* foo **' )
```

Prototype File

The file `TIBRV.rpgle` contains the prototypes of the Rendezvous calls used in the two RPG examples (which constitute only a subset of the full Rendezvous API).

Copies of `TIBRV.rpgle` exist both in `QRPGLESRC(TIBRV)` and in the IFS (at *RV_IFS_HOME*`/src/examples/rpg`).

Compiling the Samples

To compile the samples, use the following command(s):

1. Find the include file:

```
ADDLIBLE tibrv_lib
```

2. Create the module:

```
CRTRPGMOD MODULE(libr_name/TIBRVSEND) SRCFILE(tibrv_lib/QRPGLESRC)
```

3. Bind the programs:

```
CRTPGM PGM(libr_name/TIBRVSEND) BNDSRVPGM((tibrv_lib/LIBTIBRVI)(tibrv_lib/LIBTIBRV))
```

Binding Applications

It is possible that several versions of Rendezvous are installed. In this situation, you must bind and run applications with the correct version of Rendezvous. Two options are available for binding a product version.

Option A

Statically bind applications to a specific service program. For example:

```
CRTPGM PGM(yourlib/TIBRVSEND) MODULE(yourlib/TIBRVSEND) BNDSRVPGM((tibrv
_lib/LIBTIBRV) (tibrv_lib/LIBTIBRVI))
```

Option B

Dynamically bind applications with the service program, and determine the version of Rendezvous at run time. First add one of the Rendezvous product libraries to the library list, and then bind your program. For example:

```
ADDLIBLE tibrv_lib

CRTPGM PGM(yourlib/TIBRVSEND) MODULE(yourlib/TIBRVSEND) BNDSRVPGM((*LI
BL/LIBTIBRV) (*LIBL/LIBTIBRVI))
```

To verify the binding of your application, you may execute the following command (to see the library in which service program `LIBTIBRV` is found):

```
DSPPGM PGM(yourlib/TIBRVSEND) DETAIL(*SRVPGM)
```

Executing the Samples

You can execute the RPG sample programs either from the QSHELL environment or in batch.

Multi-threading is required in either environment.

Executing from the QSHELL Environment

1. Enable multi-threading (before starting the QSHELL environment):

   ```
   ADDENVVAR ENVVAR(QIBM_MULTI_THREADED) VALUE(Y) LEVEL(*SYS)
   ```

2. Start the QSHELL environment:

   ```
   STRQSH
   ```

3. Start the program:

```
system “CALL PGM(libr_name/TIBRVLISTE) PARM('7500' *DEFAULT
'myhost.my_enterprise.com:7500' 'our.subject' 'your.subject')”
```

Executing in Batch

Start the program:

```
SBMJOB CMD(CALL PGM(libr_name/TIBRVLISTE) PARM('7500' *DEFAULT
'myhost.my_enterprise.com:7500' 'our.subject' 'your.subject'))
LOG(4 0 *SECLVL) ALWMLTTHD(*YES)
```

Rendezvous Programming in RPG

Coding your own applications in RPG requires that you extend the prototype file to include all the Rendezvous calls that your application uses. Use the information in this section as a guide when you extend the prototype file, and when you code Rendezvous calls in RPG.

Parameter passing protocol in C differs from the parameter passing protocol in RPG. By default RPG passes parameters by reference, unless you explicitly code to pass by value. In contrast, C passes parameters by value (though you can explicitly supply a pointer referencing a location).

The following table guides the translation from C to RPG. Each section of the table addresses a specific type of parameter passing in C, followed by the equivalent constructs in RPG.

When C Passes a Non-String by Value	
C Prototype	`foo(int order)`
C Program Call	`int order;` `foo(order);`

RPG Prototype	`...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5`
	`d foo pr`
	`d order 10i 0 value`

RPG Program Call	`...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5`
	`D order S 10i 0 inz(37)`
	`C callp foo(order)`

When C Passes a Non-String by Reference	
C Prototype	`foo(int* order)`
C Program Call	`int order;` `foo(&order);`

RPG Prototype	`...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5`
	`d foo pr`
	`d order 10i 0`

RPG Program Call	`...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5`
	`D order S 10i 0 inz(37)`
	`C callp foo(order)`

When C Passes a String by Value	
C Prototype	`foo(char* desc)`
C Program Call	`char* desc;` `foo(desc);`

RPG Prototype	`...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5`
	`d foo pr`
	`d desc * value`

RPG Program Call	`...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5`
	`D desc S 11A`
	`/free`
	`desc = 'TIBRVLISTE' + X'00';`
	`foo(%addr(desc));`
	`/end-free`

When C Passes a String by Reference	
C Prototype	`foo(char** desc)`
C Program Call	`char* desc;` `foo(&desc);`

RPG Prototype	`...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5`
	`d foo pr`
	`d desc *`

RPG Program Call	`...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5`
	`D desc S 11A`
	`D pDesc S * inz(%addr(desc))`
	`/free`
	`desc = 'TIBRVLISTE' + X'00';`
	`foo(pDesc);`
	`/end-free`