Writing ACI Servers for the RPC-ACI Bridge in COBOL

The RPC-ACI Bridge is prepared for ACI servers written in COBOL.

This chapter covers the following topics:

- Tasks
- Data Types
- Declaring the Variables for the Data Types

Tasks

The RPC-ACI Bridge is prepared for ACI servers written in COBOL.

Writing an ACI server consists of two tasks:

- implement the Broker calls
- implement the processing of the received buffer and the response for the send buffer

Using Arrays of Groups

If your programs use arrays of groups, you have to adjust the marshalling.

To adjust the marshalling for arrays of groups

1. Use the property `entirex.rpcacibridge.marshalling` for the configuration.
2. Set the property to "cobol".

If your programs do not use arrays of groups, you do not need to set `entirex.rpcacibridge.marshalling`.

Data Types

Data Type	Format	Remarks
A<*number*> Alphanumeric	<*number*> bytes, encoding the characters	
AV[*number*] Alphanumeric variable length with maximum length	Bytes up to the end of the buffer, maximal length <*number*>	Only as last value
K<*number*> Kanji	Same as data type A	
KV[*number*] Kanji variable length with maximum length	Same as data type AV[*number*]	Only as last value
I1 Integer (small)	Sign (+, -) and 3 bytes (digits)	
I2 Integer (medium)	Sign (+, -) and 5 bytes (digits)	
I4 Integer (large)	Sign (+, -) and 10 bytes (digits)	
N<*number1*>[.*number2*] Unpacked decimal	Sign (+, -), <*number1*> bytes (digits) [*number2*] bytes (digits), no decimal point.	
NU<*number1*>[.*number2*] Unpacked decimal unsigned	<*number1*> bytes (digits) [*number2*] bytes (digits), no decimal point.	
P<*number1*>[.*number2*] Packed decimal	Sign (+, -), <*number1*> bytes (digits) [*number2*] bytes (digits), no decimal point.	
PU<*number1*>[.*number2*] Packed decimal unsigned	<*number1*> bytes (digits) [*number2*] bytes (digits), no decimal point.	
L Logical	1 Byte: X for true, all other false	
D Date	YYYYMMDD	YYYY year, MM month, DD day
T Time	YYYYMMDDhhmmssS	YYYY year, MM month, DD day, hh hour, mm minute, ss second, S tenth of a second.

Data Types not supported:

- Binary (B[n],BV, BV[n])
- Floating point (F4, F8)

Declaring the Variables for the Data Types

This section describes how to declare the variables for the data types.

Use these declarations to map the receive buffer and the send buffer to variables.

Data Type	Declaration and Marshalling
A<*number*> Alphanumeric	Declaration for receive and send buffer: PIC X(n)
AV[*number*] Alphanumeric variable length with maximum length	Declaration for receive and send buffer: PIC X(n)
K<*number*> Kanji	Declaration for receive and send buffer: PIC X(n)
KV[*number*] Kanji variable length with maximum length	Declaration for receive and send buffer: PIC X(n)
I1 Integer (small)	Declaration for receive and send buffer: PIC S9(3)
I2 Integer (medium)	Declaration for receive and send buffer: PIC S9(5)
I4 Integer (large)	Declaration for receive and send buffer: PIC S9(10)
N<*number1*>[.*number2*] Unpacked decimal	Declaration for receive and send buffer: PIC S9(*number1*)V(*number2*) SIGN LEADING SEPARATE
NU<*number1*>[.*number2*] Unsigned unpacked decimal	Declaration for receive and send buffer: PIC 9(*number1*)V(*number2*)
P<*number1*>[.*number2*] Packed decimal	Declaration for receive and send buffer: PIC S9(*number1*)V(*number2*) SIGN LEADING SEPARATE Declare local variable PIC S9(*number1*)V(*number2*) PACKED DECIMAL Move from receive buffer to local variable before computation and from local variable to send buffer afterwards.
PU<*number1*>[.*number2*] Unsigned packed decimal	Declaration for receive and send buffer: PIC 9(*number1*)V(*number2*)Declare local variable PIC 9(*number1*)V(*number2*) PACKED DECIMAL Move from receive buffer to local variable before computation and from local variable to send buffer afterwards.
L Logical	Declaration for receive and send buffer: PIC X(1)
D Date	Declaration for receive and send buffer: PIC X(8)
T Time	Declaration for receive and send buffer: PIC X(15)