Software AG IDL to COBOL Mapping

This chapter describes the specific mapping of Software AG IDL data types, groups, arrays and structures to the COBOL programming language. Please note also the remarks and hints on the IDL data types valid for all language bindings found under *Software AG IDL File*.

This chapter covers the following topics:

- Mapping IDL Data Types to COBOL Data Types
- Mapping Library Name and Alias
- Mapping Program Name and Alias
- Mapping Parameter Names
- Mapping Fixed and Unbounded Arrays
- Mapping Groups and Periodic Groups
- Mapping Structures
- Mapping the Direction Attributes In, Out, InOut
- Mapping the ALIGNED Attribute
- Calling Servers as Procedures or Functions

Mapping IDL Data Types to COBOL Data Types

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

- The metasymbols "[" and "]" surround 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	COBOL Data Type	Note
A*number*	Alphanumeric	PIC X(*number*)	
AV	Alphanumeric variable length	not supported	
AV[*number*]	Alphanumeric variable length with maximum length	PIC X(*number*)	14
B*number*	Binary	PIC X(*number*)	12

Software AG IDL	Description	COBOL Data Type	Note
BV	Binary variable length	not supported	
BV[*number*]	Binary variable length with maximum length	PIC X(*number*)	12, 14
D	Date	PIC 9(8)	1
F4	Floating point (small)	USAGE COMP-1	4
F8	Floating point (large)	USAGE COMP-2	4
I1	Integer (small)	PIC S9(2) COMP-5	10
		PIC X	9,13
I2	Integer (medium)	PIC S9(4) COMP-5	10
		PIC S9(4) BINARY	11,13
I4	Integer (large)	PIC S9(9) COMP-5	10
		PIC S9(9) BINARY	11,13
K*number*	Kanji	PIC G(*number*/2) DISPLAY-1	5
KV	Kanji variable length	not supported	
KV[*number*]	Kanji variable length with maximum length	PIC G(*number*/2 DISPLAY-1)	5, 14
L	Logical	PIC X	6,7
N*number1*[.*number2*]	Unpacked decimal	PIC S9(*number1*) [V(*number2*)]	2
NU*number1*[.*number2*]	Unpacked decimal unsigned	PIC 9(*number1*) [V(*number2*)]	2
P*number1*[.*number2*]	Packed decimal	PIC S9(*number1*) [V(*number2*)] PACKED-DECIMAL	2
PU*number1*[.*number2*]	Packed decimal unsigned	PIC 9(*number1*) [V(*number2*)] PACKED-DECIMAL	2
T	Time	PIC 9(15)	3
U*number*	Unicode	PIC N(*number*) NATIONAL	8
UV	Unicode variable length	not supported	

Software AG IDL	Description	COBOL Data Type	Note
UV*number*	Unicode variable length with maximum length	PIC N(*number*) NATIONAL	8, 14

See also the hints and restrictions under *Software AG IDL File* valid for all language bindings.

Notes:

1. The date corresponds to the format PIC 9(8). The value contained has the form YYYYMMDD. This form corresponds to COBOL DATE functions. This is an IBM extension of COBOL85 standard.

2. For COBOL, the total number of digits (*number1*+*number2*) is lower than the maximum of 99 that EntireX supports. See *IDL Data Types*. It varies by operating system and COBOL compiler. To enable more total number of digits than 18, a compiler directive (option) may be required.

   Under **z/OS**:

   - The total number of digits (*number1*+*number2*) is restricted to 31 digits.
   - The compiler option AR(E) is generated into the client interface objects and server skeletons if more than 18 digits are defined in the IDL.

   Under **Micro Focus**:

   - The total number of digits (*number1*+*number2*) is restricted to 38 digits.
   - The compiler option INTLEVEL"4" is generated into the client interface objects and server skeletons if more than 18 digits are defined in the IDL.

   Under **BS2000/OSD**:

   - The total number of digits (*number1*+*number2*) is restricted to 31 digits.

   Under **z/VSE**:

   - The total number of digits (*number1*+*number2*) is restricted to 18 digits.

   Under all other operating systems or compilers:

   - Refer to your COBOL compiler documentation to see whether compiler directives or options exist.

   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.

3. The time corresponds to the format PIC 9(15). The value contained has the form YYYYMMDDHHIISST. This form corresponds to COBOL DATE/TIME functions.

4. When floating-point data types are used, rounding errors can occur, so that the values of senders and receivers might differ slightly.

5. The length for IDL data type is given in bytes. For COBOL the length is in DBCS characters (2 bytes). IDL data type K is not supported under BS2000/OSD because Fujitsu Siemens compilers do not support DBCS.

6. To inspect the Boolean value of a data item of IDL type Logical, you can specify PIC X followed by condition names (similar code is generated for scalar logical IDL types):

```
level-number data-name PIC X.
88           data-name-false value X'00'.
88           data-name-true  value X'01' thru X'FF'.
```

   Under **IBM i,**

   The SYMBOLIC CHARACTERS clause in the SPECIAL-NAMES paragraph is not supported. The following COBOL statements demonstrate how you can define alternatively a character, named HEX-00, with a value of hexadecimal zero to be used for comparison:

```
WORKING-STORAGE SECTION.
01  HEX-00-B           PIC 9(4) BINARY VALUE 0.
01  HEX-00-H REDEFINES HEX-00-B.
   02  FILLER         PIC X.
   02  HEX-00         PIC X.
```

7. To set the Boolean value of a Logical data item, specify the following hexadecimal values in a one-byte data field (e.g. defined as PIC X.):

   - Case False: Move X'00' to *data-name*.
   - Case True: Move X'01' to *data-name*.

8. The length is given in Unicode code units following the Unicode standard UTF-16.

   Under **z/OS and IBM Compiler:**

   - Unicode requires the IBM Enterprise compiler.
   - Unicode is represented in UTF-16 big-endian format (CCSID 1200).

   Under **BS2000/OSD:**

   - Unicode requires a compiler that supports COBOL data type NATIONAL. See *BS2000/OSD Prerequisites*.
   - Unicode is represented in UTF-16 big-endian format.

   Under **Micro Focus** (UNIX and Windows):

   - Set the compiler directive NSYMBOL"NATIONAL".
   - **For clients**
     Unicode can be represented in UTF-16 big-endian format (compiler directive UNICODE(PORTABLE)) or machine-dependent endianness UTF-16 big or little endian (compiler directive UNICODE(NATIVE)).

- **For servers**
  Unicode can be represented in UTF-16 machine-dependent endianness (big or little endian) format only. `UNICODE(PORTABLE)` is not supported.

Under all other operating systems or compilers:

- Refer to your COBOL compiler documentation.

9. COBOL for operating systems z/OS, z/VSE, BS2000/OSD and IBM i does not have a corresponding data type for a compatible I1 mapping. The mapping to COBOL `PIC X` data type should be seen as a `FILLER` variable. If including an I1 data type into the interface is required, it is your responsibility as application developer to process the content of this parameter provided (during receive) and expected (during send) correctly. Negative values are given as the two's complement binary number.

10. Supported for Micro Focus COBOL for operating systems UNIX and Windows only.

11. The value range for COBOL data type `BINARY` on z/OS, z/VSE, BS2000/OSD and IBM i depends on the COBOL compiler settings:

    - With COBOL 85 standard, the mapped COBOL data type `BINARY` is more restrictive than the IDL data types I2 and I4. See *IDL Data Types*. This means that COBOL RPC clients cannot send (and COBOL RPC servers cannot return) the full value range defined by the IDL types I2 and I4. On the other hand, COBOL RPC clients and COBOL RPC servers may receive a value range (from a non-COBOL RPC partner) outside of the value range of your COBOL data type.

    - *Without* COBOL 85 standard, the value range of the COBOL data type `BINARY` depends on the binary field size, thus matches the IDL data type exactly. In this case, there are no restriction regarding value ranges.

    - To match the value range of IDL type I2 and I4 exactly, depending on the operating system, the following compiler directive (option) is generated into the client interface objects and server skeletons:

      Under **z/OS** and **z/VSE**:

      - the IBM compiler option `TRUNC(BIN)`

      Under all other operating systems or compilers:

      - refer to your COBOL compiler documentation to see whether compiler directives or options exist.

12. COBOL does not have a corresponding data type for a compatible B/BV mapping. Thus the mapping is to COBOL `PIC X` data type. EntireX RPC transports the (binary) data as it is: no character translation or conversion will be performed.

13. Supported for operating systems z/OS, z/VSE, BS2000/OSD and IBM i only.

14. With variable length fields with maximum (AV*n*, BV*n*, KV*n* and UV*n*), mapping to endpoints with a concept of real string types - such as Java, .NET, C, XML, Web services etc. - is straightforward. The transfer of data in the RPC data stream depends on the actual length of the string and not the field size, as seen in COBOL. For the COBOL side, the actual content length of such fields is determined using a trim mechanism. For AV*n*, all trailing `SPACE`s are ignored before send. After receive, the

content is padded with trailing SPACEs up to the COBOL field size. For BV*n*, HEX ZERO is used instead of SPACE; for UV*n*, Unicode code point U+0020. See also the notes under *IDL Data Types* under *Software AG IDL File* in the IDL Editor documentation.

Mapping Library Name and Alias

Client Side

The IDL library name as specified in the IDL file (there is no 8-character limitation) is sent from a client to the server. Special characters are not replaced. The library alias is neither sent to the server nor used for other purposes on the COBOL client side.

Server Side

If you are using a so-called server mapping file, the target COBOL server program is located with the help of this file. A server mapping file is an EntireX Workbench file with extension .svm or .cvm. See *Server Mapping Files for COBOL*. See also *Locating and Calling the Target Server* under z/OS (CICS, Batch, IMS) | Micro Focus | BS2000/OSD | z/VSE (CICS, Batch) | IBM i.

If you are *not* using a server mapping file, the IDL library name as specified in the IDL file is ignored.

Mapping Program Name and Alias

Client Side

The IDL program name as specified in the IDL file (there is no 8-character limitation) is sent from a client to the server. Special characters are not replaced. The program alias is not sent to the server, but during wrapping it is used to derive the suggestion for the source file names of the client interface objects (COBOL subprograms, copybooks) instead of using the IDL program names, see *Customize Automatically Generated Client Names*.

Server Side

If you are using a so-called server mapping file, the target COBOL server program is located with the help of this file. A server mapping file is an EntireX Workbench file with extension .svm or .cvm. See *Server Mapping Files for COBOL*. This provides the following advantages:

- IDL program names are not limited to 8 characters and do not have to match the target COBOL server program names.
- Target COBOL server program names (COBOL subprograms) can be customized during wrapping. See *Customize Automatically Generated Server Names*.

If you are *not* using a server mapping file, the target COBOL server program must match the IDL program name. In this case:

- The length of the IDL program names is limited by your COBOL system (often 8 characters).

- The set of allowed characters for IDL program names is restricted by your COBOL system and the underlying file system.

It is your responsibility as application developer to ensure that these requirements are met. See *Locating and Calling the Target Server* under z/OS (CICS, Batch, IMS) | Micro Focus | BS2000/OSD | z/VSE (CICS, Batch) | IBM i.

Mapping Parameter Names

The parameter names, as given in the `parameter-data-definition` of the IDL file, are mapped to fields within the `LINKAGE` section of the generated COBOL client interface objects and COBOL server skeletons.

When building fields within the `LINKAGE` section, the special characters '#', '$', '&', '+', '-', '.', '/', '@' and '_', allowed within names of parameters, are mapped to the character hyphen '-' valid for COBOL names. Example:

`HU$GO` results in `HU-GO`

Trailing and preceding special characters are also removed. Example:

`#HUGO$` results in `HUGO`

Subsequent special characters are replaced by one hyphen. Example:

`HU$#$GO` results in `HU-GO`

If the parameter name starts with a digit, e.g. '`1`', it is prefixed with the character '`P`'. Example:

`1HUGO` results in `P1HUGO`

Mapping Fixed and Unbounded Arrays

Client and Server Side

- Fixed arrays within the IDL file are mapped to fixed COBOL 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 IDL file and refer to `fixed-bound-array-index`.
- For clients on all operating systems, and for servers on the operating systems z/OS, BS2000/OSD, z/VSE, UNIX and Windows for Micro Focus COBOL, IDL unbounded arrays with a maximum are mapped to COBOL tables with the `DEPENDING ON` clause. See *Tables with Variable Size - `DEPENDING ON` Clause* under *COBOL to IDL Mapping* in the IDL Extractor for COBOL documentation. Note the following:
  - The `from-value` of the `DEPENDING ON` clause is always 1.
  - ODO objects for justification of the number of occurrences are generated into the client interface objects and server skeletons.

  - When a 2/3 dimensional unbounded array is received from a partner, all vectors of the second dimension must have the same length, i.e. the array forms a rectangle. The same applies to the third dimension (all vectors must have the same length), the array forms a cuboid. If these rules are violated, unexpected behavior occurs. For illustration, see picture under *array-definition* under *Software AG IDL Grammar* in the IDL Editor documentation.

  - Sending a 2/3 dimensional unbounded array to a partner violating the rule above is not possible: COBOL does not allow you to set vector lengths differently.

- For servers on the operating system IBM i, IDL unbounded arrays with a maximum are mapped to fixed COBOL tables. On the reply, the number of occurrences is determined by `NULL` value contents. Occurrences with null values are not sent back to the calling RPC client.

- Unbounded arrays without a maximum are *not* supported.

Mapping Groups and Periodic Groups

Client and Server Side

- Groups within the IDL file are mapped to COBOL structures using level numbers. See the `group-parameter-definition` for the syntax on how to describe groups within the IDL file.

- For clients on all operating systems and for servers on the operating systems z/OS, BS2000/OSD, z/VSE, UNIX and Windows for Micro Focus COBOL, IDL with unbounded groups with a maximum:

  - the same applies as for unbounded arrays, see *Mapping Fixed and Unbounded Arrays*

  - if unbounded groups are nested, and depending on your target COBOL compiler,

    - they may not be supported (e.g. BS2000/OSD).

    - there is a restriction on the number of indices. Most COBOL compiler support 7 indices as a maximum.

      The EntireX Workbench generates the COBOL interface objects and server (skeletons) without considering restrictions of the target COBOL compiler. See your COBOL compiler documentation for possibilities to work round the restrictions, for example using compiler switches or compiler options.

- For server on the operating system IBM i, Software AG IDL unbounded groups with a maximum are mapped to fixed COBOL tables. On the reply the number of occurrences is determined by NULL value contents. Occurrences with null values are not sent back to the calling RPC client.

- Unbounded groups without a maximum are not supported.

Mapping Structures

Client and Server Side

Structures within the IDL file are dissolved at the location where they are used. They are mapped to COBOL structures like groups. See the `structure-definition` under *Software AG IDL Grammar* for the syntax on how to describe structures within the IDL file.

Mapping the Direction Attributes In, Out, InOut

The IDL syntax allows you to define parameters as In parameters, Out parameters, or InOut parameters (which is the default if nothing is specified). See the `attribute-list` under *Software AG IDL Grammar* for the syntax on how to describe attributes within the IDL file and refer to `direction-attribute`.

Client Side

This direction specification is reflected in the generated COBOL interface object as follows:

- Direction attributes do not change the COBOL 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 IDL file and refer to `direction-attribute`.

Server Side

If you are using a server mapping file, the RPC server considers the direction attribute found in the server mapping file. A server mapping file is an EntireX Workbench file with extension .svm or .cvm. See *Server Mapping Files for COBOL*.

If your RPC server is generated with a previous version of EntireX without a server mapping file, the RPC server considers the direction attribute sent from any RPC client, for example Java, DCOM, C, COBOL, .NET, XML and PL/I.

Mapping the ALIGNED Attribute

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

Client and Server Side

This attribute corresponds to the SYNCHRONIZED clause. If it is specified, data will be mapped according to the following rules:

Software AG IDL	COBOL Data Type	Alignment	Notes
F4	USAGE COMP-1 SYNC	+4	1
F8	USAGE COMP-2 SYNC	+8	1
I2	PIC S9(4) BINARY SYNC	+2	1
I4	PIC S9(8) BINARY SYNC	+4	1

Notes:

1. On IBM i, specify the compiler option *SYNC in the commands CRTCBLMOD or CRTBNDCBL for the usage of the SYNCHRONIZED clause.

Calling Servers as Procedures or Functions

Client and Server Side

The COBOL 85 standard does not support a concept of functions like the programming languages C or PL/I. Any Software AG IDL program definition is mapped to a COBOL program. See *Mapping Program Name and Alias*.