Software AG IDL File

A Software AG IDL file contains definitions of the interface between client and server. The IDL file is used by Software AG wrappers to generate RPC clients, RPC servers and tester etc. on the basis of these definitions. The IDL file can be edited by the IDL Editor provided by plug-ins for Eclipse.

This document contains a descriptive introduction to IDL files. The syntax of IDL files in a formal notation is given under *Software AG IDL Grammar*. This chapter covers the following topics:

- Introduction to the IDL File
- IDL Data Types
- Fixed and Unbounded Arrays
- Rules for Coding IDL Files
- Rules for Coding Group and Parameter Names
- Rules for Coding Library, Library Alias, Program, Program Alias and Structure Names
- Context Menu

Introduction to the IDL File

The IDL's syntax looks similar to a Software AG Natural parameter data definition statement.

```
Library 'EXAMPLE' Is
        Program 'CALC' 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
```

The syntax is described in a formal notation under *Software AG IDL Grammar*.

IDL Data Types

The table below uses the following metasymbols and informal terms 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.

Type and Length	Description	Example	See Notes
A*number*	Alphanumeric	A100	1, 2, 7, 16, 19

Type and Length	Description	Example	See Notes
AV	Alphanumeric variable length	AV	1, 2, 7, 16, 19, 20
AV*number*	Alphanumeric variable length with maximum length	AV100	1, 2, 7, 16, 19, 20
B*number*	Binary	B10	1, 2, 14
BV	Binary variable length	BV	1, 2, 14, 20
BV*number*	Binary variable length with maximum length	BV128	1, 2, 14, 20
D	Date	D	3, 4, 13
F4	Floating point (small)	F4	11, 13, 15
F8	Floating point (large)	F8	12, 13, 15
I1	Integer (small)	I1	8
I2	Integer (medium)	I2	9
I4	Integer (large)	I4	10
K*number*	Kanji	K20	1, 2, 7, 16, 17, 19
KV	Kanji variable length	KV	1, 2, 7, 16, 17, 19, 20
KV*number*	Kanji variable length with maximum length	KV200	1, 2, 7, 16, 17, 19, 20
L	Logical	L	3, 13
N*number1*[.*number2*]	Unpacked decimal	N8 or N8.2	6
NU*number1*[.*number2*]	Unpacked decimal unsigned	NU2 or NU6.2	6
P*number1*[.*number2*]	Packed decimal	P12 or P10.3	6
PU*number1*[.*number2*]	Packed decimal unsigned	PU3 or PU4.2	6
T	Time	T	3, 5, 13
U*number*	Unicode	U100	2, 18
UV	Unicode variable length	UV	2, 18, 20
UV*number*	Unicode variable length with maximum length	UV200	2, 18, 20

Note that equivalents of the data types are not necessarily supported in every target programming language environment. Also, value ranges of the mapped data type can differ. See *Mapping IDL Data Types* to target language environment C | CL | COBOL | DCOM | .NET | Java | Natural | PL/I | RPG.

Notes:

1. There is, however, an absolute limit (1 GB) which cannot be exceeded.
2. The maximum length you can specify depends on your hardware and software configuration (apart from this product).
3. The length is implicit and must not be specified.
4. The supported range is from 1.1.0001 up to 31.12.9999. Dates BC (before the birth of Christ) are not supported.

   It is also possible to transfer 1.1.0000 as a value. This is a special value (because there is no year 0) and denotes "no date" is given. The no date value is the internal state of a `#DATE` variable (Natural type D) after a `RESET #DATE` is executed within Natural programs. The target language environment determines how ’no date’ is handled.

   See the notes under data type D in the section *Mapping Software AG IDL Data Types* to the target language environment C | Java | .NET.
5. The data type T has two different meanings:

   - A time-only meaning, which transfers a time without a date. The time-only meaning always uses the invalid date 1.1.000 for the date part. The time part has a value range from 00:00:00.0 to 23:59:59.9. This time-only meaning is not supported.
   - A timestamp meaning, consisting of a date and time.

     The supported range is from 1.1.0001 0:00:00.0 up to 31.12.9999 23:59:59.9. Dates BC (before the birth of Christ) are not supported.

     It is also possible to transfer 1.1.0000 0:00:00.0 as a value. This is a special value (because there is no year 0) and denotes "no time" is given. The "no time" value is the internal state of a `#TIME` (Natural type T) variable after a `RESET #TIME` is executed within Natural programs. The target language environment determines how "no time" is handled.

     See the notes under data type T in the section *Mapping Software AG IDL Data Types* to the target language C | Java | .NET.
6. The term *`number1`*[.*`number2`*] describes the number as it is: The first number is the number of digits before the decimal point and the second number is the number of digits after the decimal point. The total number of digits (*`number1`*+*`number2`*) must not exceed 99. Depending on your target programming language, the total number of digits can be more restricted.

   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.
7. The length is given in bytes, not in number of characters.

8. The valid integer range is from -128 up to +127.
9. The valid integer range is from -32768 up to +32767.
10. The valid integer range is from -2147483648 up to +2147483647.
11. The following term restricts the valid range which can be transferred from -n.nnnnnn+Enn up to +n.nnnnnn+Enn. A mantissa of 7 decimal digits and an exponent of 2 decimal digits.
12. The following term restricts the valid range which can be transferred from -n.nnnnnnnnnnnnnnn+Enn up to +n.nnnnnnnnnnnnnnn+Enn. A mantissa of 16 decimal digits and an exponent of 2 decimal digits.
13. Valid values are TRUE and FALSE.
14. The length is given in bytes.
15. When using floating-point values, rounding errors can occur when converting to the target language environment. Thus, values from sender and receiver might differ slightly.
16. In environments where multibyte, double-byte or other complex codepages are used, alphanumeric data may increase or decrease during conversion. Thus, to match the field length restriction given by the IDL types A and AV with maximum length, data must be truncated, otherwise unpredictable results will occur. The most popular internationalization approach *ICU Conversion* with `CONVERSION=SAGTRPC` takes care of data increase/decrease.

    We recommend always using SAGTRPC for RPC data streams. *Conversion with Multibyte, Double-byte and other Complex Codepages* will always be correct, and *Conversion with Single-byte Codepages* is also efficient because SAGTRPC detects single-byte codepages automatically. See *Conversion Details*.

    See also *Configuring ICU Conversion* under z/OS | UNIX | Windows | BS2000/OSD | z/VSE.
17. In environments that use EBCDIC stateful codepages, encoded with escape technique (SI/SO bytes), and where the most popular internationalization approach *ICU Conversion* with `CONVERSION=SAGTRPC` is used, the IDL types K and KV fields allow you to transfer double-byte data without SO and SI bytes. This feature is designed for use in Asian countries. For more information see *Conversion with Multibyte, Double-byte and other Complex Codepages*.
18. The length is given in 2-byte Unicode code units following the Unicode standard. UTF-16. The maximum length is restricted to 805306367 2-byte code units.

    Depending on your target environment and target programming language, the mapping may follow a different Unicode standard, for example UTF-32.
19. If *SAGTRPC User Exit* is used as the internationalization approach, the handling of the different IDL types depends on the implementation of the SAGTRPC user exit. This is your responsibility as user. See *Writing SAGTRPC User Exits* under z/OS | UNIX | Windows.
20. Variable-length (e.g. AV, AV*n*) fields are transferred in the RPC data stream in the length specified. A defined maximum in the IDL file limits the number of elements that can be transferred.

Variable-length fields with maximum (e.g. AV*n*) are important for connections to endpoints that have no concept of variable-length data, such as COBOL (see *Software AG IDL to COBOL Mapping*) and PL/I (see *Software AG IDL to PL/I Mapping*).

Fixed and Unbounded Arrays

A fixed array is transferred in the RPC data stream with all its elements.

With an unbounded array, the current number of elements and their contents are transferred in the RPC data stream. A defined maximum in the IDL file limits the number of elements that can be transferred.

For the formal syntax of arrays, refer to *`array-definition`* under *Software AG IDL Grammar*.

Unbounded arrays with a maximum are important for connections to COBOL, which supports a similar concept with the `OCCURS DEPENDING ON` clause. See *Tables with Variable Size - `DEPENDING ON` Clause* under *COBOL to IDL Mapping* in the IDL Extractor for COBOL documentation.

Rules for Coding IDL Files

1. Statements and their lexical entities can begin in any column and are separated by any number of whitespace characters: blank, new line carriage return, horizontal tab, and form feed.
2. The maximum line length allowed in an IDL file is 256 characters.
3. Comments can be entered in the following ways:
   - If the entire line is to be used for a user comment, enter an asterisk or a slash and an asterisk in columns 1 and 2 of the line:

     ```
     *    USER COMMENT
     /*   USER COMMENT
     ```

   - If only the latter part of a line is to be used for a user comment, enter an asterisk or slash asterisk.

     ```
      1 NAME    (A20)      * USER COMMENT
      1 NUMBER  (A15)     /* USER COMMENT
     ```

Rules for Coding Group and Parameter Names

Group and parameter names

1. can be defined with the following characters:
   - characters: a to z
   - characters: A to Z
   - digits: 0 to 9 (a digit must not be the first character)

- special characters: - _ $ # & @ + /

other characters are not allowed.

2. are limited to a maximum length of 31 characters
3. are not allowed to be the same as a valid type-length specification.

   For example:

   ```
   1 P1 (P1) In Out
   ```

   is invalid and will cause an error because the name `P1` is identical to the type-length `P1`.

4. must adhere to the rules of the target programming language, for example to permitted special characters or reserved keywords.
5. cannot be defined as the following reserved names:

   `ALIGNED`, `CALLNAT`, `DATA`, `DEFINE`, `END-DEFINE`, `IMS`, `IN`, `INOUT`, `IS`, `LIBRARY`, `OUT`, `PARAMETER`, `PROGRAM`, `RCODE`, `STRUCT`, `VERSION`.

6. must be unique and must not conflict with those of the target programming language, see the following portion of an IDL file

   ```
   Define Data Parameter
   1 AA   (I2)
   1 AA   (I4)
   1 long (I4)
   End-define
   ```

   and the output generated with the client.tpl as the template for target language C:

   ```
   short int AA;
   long      AA;      /*erroneous, double declaration*/
   long      long;    /*erroneous, double declaration*/
   ```

   The ambiguous declaration of `AA` and `long` is passed unchecked and the stub will be generated. As you can see, this is not valid C syntax.

Rules for Coding Library, Library Alias, Program, Program Alias and Structure Names

The following rules apply to library, library alias, program, program alias and structure names:

1. Names are restricted by length. Library, library alias, program and program alias are restricted to a maximum length of 128 characters. A structure name is restricted to a maximum length of 31 characters.
2. Names must adhere to the rules of the target programming language, for example regarding permitted special characters or reserved keywords.

3. Names should not start with the prefix "SAG". The prefix "SAG" is used within the delivered IDL files. See *Change RPC Password by Wrappers and RPC Clients* and *Command and Info Services IDLs* for more information.

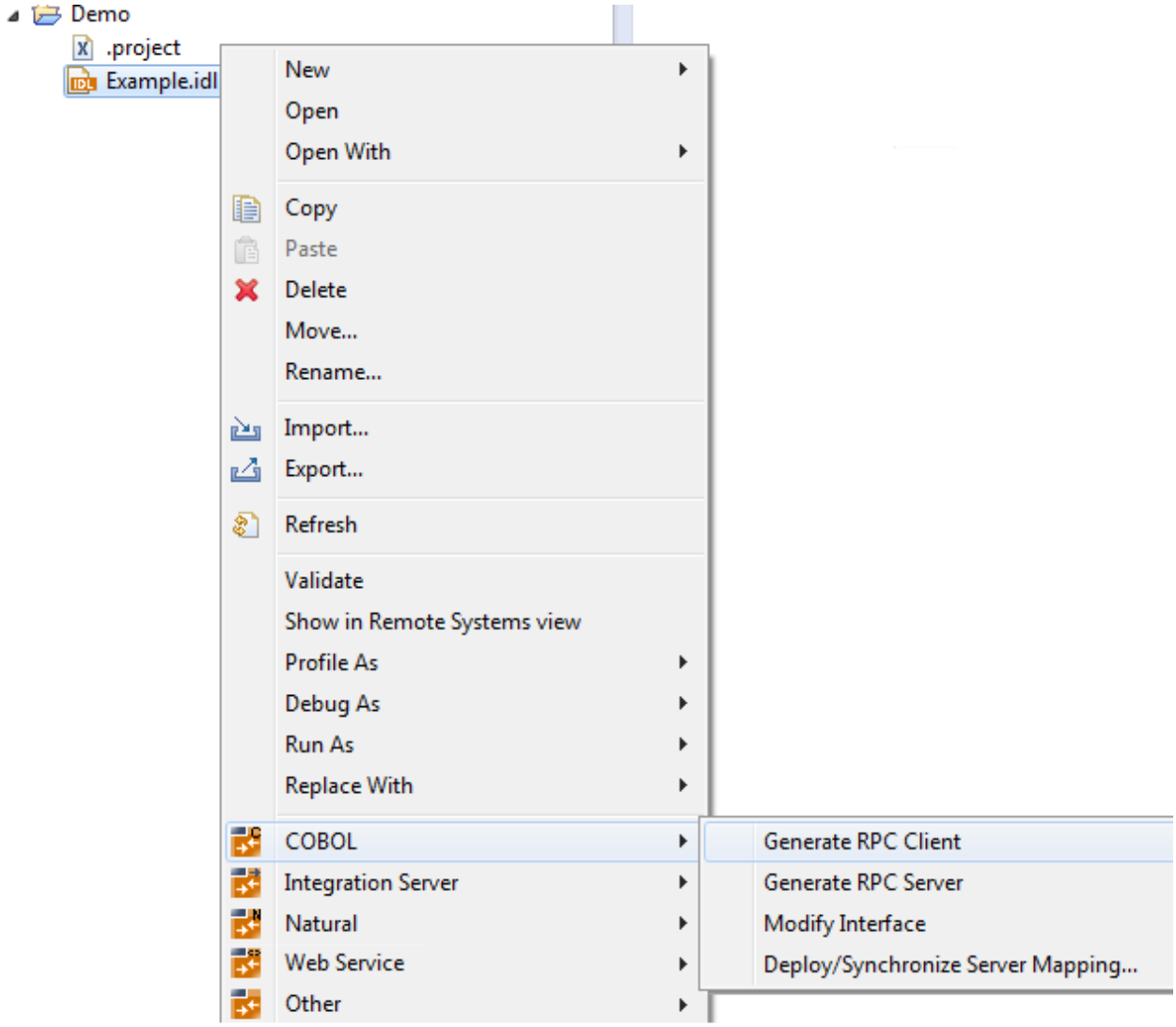
4. Names must be unique and different within the IDL file after conversion of the name to lowercase or uppercase characters. You cannot use the same name for a library, library alias, program, program alias and structure.

   Example: The following names are not allowed within an IDL file:

   - `MYLIBRARY` and `MyLibrary`
   - `CALC` and `Calc`
   - `MYSTRUCTURE` and `mystructure`

Context Menu

The context menu for IDL files in the EntireX Workbench has the most commonly used target environments - COBOL, Integration Server, Natural and Web Services - on the first menu level. Under **Other** you can specify additional supported targets such as C or Java, start the IDL Tester or refactor the IDL file. See example for COBOL below:


Demo
.project
Example.idl
New
Open
Open With
Copy
Paste
Delete
Move...
Rename...
Import...
Export...
Refresh
Validate
Show in Remote Systems view
Profile As
Debug As
Run As
Replace With
COBOL
Integration Server
Natural
Web Service
Other
Generate RPC Client
Generate RPC Server
Modify Interface
Deploy/Synchronize Server Mapping...