Natural to IDL Mapping

This chapter describes how Natural data types are mapped to Software AG IDL files by the Software AG IDL Extractor for Natural and covers the following topics:

- Mapping Natural Data Types to Software AG IDL
- Redesigning the Extracted Interface
- Extracting the IDL Library Name
- Extracting the IDL Program Name
- Extracting IDL Parameter Names
- Extracting IDL Directions (IN,OUT,INOUT)
- Extracting Natural REDEFINES
- Extracting Multiple Interfaces
- Extracting Natural Arrays, Groups, X-Arrays and Variable Arrays
- Extracting Natural Structure Information (IDL Levels)
- Extracting Parameters defined with OPTIONAL
- Setting Natural Parameters to Constants
- Suppressing Natural Parameters
- Renaming a Program

For more information on Natural syntax, refer to the Natural documentation.

Mapping Natural Data Types to Software AG IDL

The IDL Extractor for Natural maps the following subset of Natural data types to Software AG IDL data types.

The following metasymbols and informal terms are used for the IDL in the table below.

- The metasymbols "[" and "]" surround optional lexical entities
- The informal terms *n* and *m* are sequences of numeric characters, for example 123.

Natural Data Type	Software AG IDL Data Type	Description
A*number*	A*n*	Alphanumeric
A DYNAMIC	AV*n*	Alphanumeric variable length
B*number*	B*number*	Binary
B DYNAMIC	BV	Binary variable length
C	not supported	
D	D	Date
F4	F4	Floating point (small)
F8	F8	Floating point (large)
I1	I1	Integer (small)
I2	I2	Integer (medium)
I4	I4	Integer (large)
L	L	Logical
N*number*[.*number*]	N*number*[.*number*]	Unpacked decimal
P*number*[.*number*]	P*number*[.*number*]	Packed decimal
T	T	Time
U*number*	U*number*	Unicode
U DYNAMIC	UV	Unicode variable length

Redesigning the Extracted Interface

The IDL Extractor for Natural allows you to design the interface to your Natural subprogram (CALLNAT). This includes

- *Extracting Multiple Interfaces*
- *Extracting Natural REDEFINES*
- *Extracting IDL Directions (IN,OUT,INOUT)*
- *Setting Natural Parameters to Constants*
- *Suppressing Natural Parameters*

See *Step 6: Redesign the Interface for Natural Subprograms (Optional)* for more information.

Extracting the IDL Library Name

The Natural library from where Natural programs are extracted is used as the IDL library name. See library-definition under *Software AG IDL Grammar*.

Extracting the IDL Program Name

The Natural program name is used as the IDL program name, see `program-definition` under *Software AG IDL Grammar*.

Extracting IDL Parameter Names

For **source extractions**, Natural parameter names are kept and used as IDL parameters, see `simple-parameter-definition` and `group-parameter-definition` under *Software AG IDL Grammar*.

For **object extractions**, Natural programs must be compiled (cataloged) with the compiler option `SYMGEN=ON` to keep original Natural parameter names. Otherwise, generic parameter names are generated (`PARAMETER-1`, `PARAMETER-2`, etc.).

In **Select Natural Sources** (see *Step 3: Select the Natural Subprograms from NaturalONE Project* if you are extracting from NaturalONE projects or *Step 5: Select Natural Subprograms from RPC Environment* if you are extracting from a Natural RPC environment), you can choose special characters (`$`, `#`, `&`, `@`, `/`) in Natural parameter names to be replaced by underscores. See *Rules for Coding Group and Parameter Names*.

Extracting IDL Directions (IN,OUT,INOUT)

In most Natural subprograms, parameters have no specification for a direction. Missing a direction is unproblematic for local calls. For remote RPC calls, however, specifying the direction helps to reduce data sizes.

If you redesign the interface, you can define IDL directions in *Step 6: Redesign the Interface for Natural Subprograms (Optional)* using the mapping operations **Map to In**, **Map to Out**, **Map to InOut**.

Otherwise, IDL directions can be inserted at top-level parameters (level 1) using a Natural line comment in the Natural subprogram (`CALLNAT`) interface definition (`DEFINE DATA PARAMETER`), example:

```
DEFINE DATA PARAMETER
1 #IN-FIELD-1            (P9)  /* IN
1 #OUT-FIELD-1           (P9)  /* OUT
1 #INOUT-FIELD-1         (P9)  /* INOUT
1 #INOUT-FIELD-2         (P9)
1 #IN-GROUP-1                  /* IN
  2 #IN-GROUP-FIELD-1    (A10)
1 #OUT-GROUP-1                 /* OUT
  2 #OUT-GROUP-FIELD-1   (A10)
1 #INOUT-GROUP-1               /* INOUT
  2 #INOUT-GROUP-FIELD-1 (A10)
1 #INOUT-GROUP-2
  2 #INOUT-GROUP-FIELD-2 (A10)
1 #INOUT-GROUP-3
  2 #INOUT-GROUP-FIELD-3 (A10) /* OUT
END-DEFINE
```

If no direction is specified (such as in `#INOUT-FIELD-2` and `#INOUT-GROUP-2` in the example above), the default direction INOUT applies.

Specifications on a level greater than 1 (such as `#INOUT-GROUP-FIELD-3` in the example above) are ignored. Note that in IDL directions are specified on top-level fields (level 1), see `attribute-list` under *Software AG IDL Grammar*.

Specifications on IDL directions are only considered when extracting from a source. If you are extracting from an object (compiled), as described in *Step 5: Select Natural Subprograms from RPC Environment*, the default direction `INOUT` always applies.

Extracting Natural REDEFINES

A redefinition is a second parameter layout of the same memory portion. The parameter `#BASE-FIELD` is redefined by the fields `FILLER-1` thru `R-P3-01`.

```
DEFINE DATA PARAMETER
1 #BASE-FIELD            (A161)
1 REDEFINE #BASE-FIELD
  2 FILLER-1             (A4)
  2 FILLER-2             (A60)
  2 R-P1-01              (A1)
  2 R-P2-01              (A10)
  2 R-P3-01              (I4)
END-DEFINE
```

With the extractor wizard you can select a single redefine path for IDL usage (here the fields `FILLER-1` thru `R-P3-01`) if you redesign the interface. See *An Example for Extracting Natural `REDEFINES`* and *Step 6: Redesign the Interface for Natural Subprograms (Optional)*.

Extracting Multiple Interfaces

Legacy Natural subprograms often implement multiple functions in a single Natural subprogram. The function executed is often controlled by a so-called function code or operation-code field. See *An Example for Extracting Multiple Interfaces*.

With the extractor wizard you can extract the functions from the server as separate interfaces (IDL programs). In this way, the legacy server with a single physical interface can be

- turned into a web service with operations, where the legacy functions match operations.
- called with an object-oriented wrapper such as the *Java Wrapper*, the *.NET Wrapper* or the *DCOM Wrapper*, where the legacy functions match methods.

Note that every function in the Natural subprogram may have a different interface described with `REDEFINE` syntax. Therefore, multiple interface extraction is often combined with *Extracting Natural `REDEFINES`*.

For more information, see *Step 6: Redesign the Interface for Natural Subprograms (Optional)*.

Extracting Natural Arrays, Groups, X-Arrays and Variable Arrays

This section describes IDL mapping for Natural arrays and groups:

Arrays and Groups with Fixed upper Limits

Ordinary Natural arrays and groups with fixed/bound upper limits are mapped to Software AG IDL fixed-bound-array definitions, see *array-definition* under *Software AG IDL Grammar* in the IDL Editor documentation.

Natural syntax example:

```
DEFINE DATA PARAMETER
1 #ARRAY1   (I4/1:10) /* lower bound is fixed at 1, upper bound is 10
1 #ARRAY2   (I4/10)   /* shortcut for (A5/1:10)
1 #GROUP1   (10)
  2 #FIELD1 (I2)
  2 #FIELD2 (A10)
. . .
END-DEFINE
```

X-Arrays and X-Groups

For X-arrays (eXtensible arrays) the number of occurrences is flexible at runtime. The number of occurences can be resized, i.e. increased or reduced. It is defined by specifying an asterisk (*) for index bounds.

Natural syntax example:

```
DEFINE DATA LOCAL
1 #X-ARRAY1 (A5/1:*) /* lower bound is fixed, upper bound is variable
1 #X-ARRAY2 (A5/*)   /* shortcut for (A5/1:*)
. . .
END-DEFINE
```

Natural X-arrays are mapped to Software AG IDL unbounded-array definitions, see `array-definition`.

Natural X-arrays with variable lower bounds are *not* supported by Software AG RPC technology, example:

```
DEFINE DATA PARAMETER
1 #X-ARRAY1 (A5/*:10) /* lower bound is variable, upper bound is fixed
. . .
END-DEFINE
```

Variable Arrays and Variable Groups

In a Natural parameter data area (PDA), you can specify an array or group with a variable number of occurrences. This is done with the index notation `1:V`. The maximum number of occurrences for such an array is either passed to the subprogram using an extra parameter such as `#ARRAY1-LIMIT` (see example below), or it can be accessed using the system variable `*OCCURRENCE`.

Natural syntax example:

```
DEFINE DATA PARAMETER
1 #ARRAY1-LIMIT (I4) /* extra parameter to pass the upper limit
1 #ARRAY1       (I4/1:V)
. . .
END-DEFINE
```

Natural variable arrays are mapped to Software AG IDL unbounded-array definitions, see `array-definition`.

If the Natural server program uses a separate parameter such as `#ARRAY1-LIMIT` (see the example above) instead of `*OCCURRENCE` to determine the upper bound limit, it is required to extract this extra parameter, too. During runtime, it is also required to specify the number of occurences in a calling RPC client.

In a Natural server program, Natural variable arrays

- cannot be resized for direction INOUT, which means you can only reply the same number of occurrences to the RPC client.
- cannot be used for direction OUT either, because they cannot be created (instantiated). You may get error 20050031 during extraction.

Arrays and Groups with Mixed Dimensions (X, Variable and Fixed)

Natural arrays and groups with a mixture of fixed variable and eXtensible dimensions are *not* supported by Software AG RPC technolgy, example:

```
DEFINE DATA PARAMETER
1 #ARRAY1   (I4/1:10,1:*) /* first dimension fixed and second eXtensible
1 #ARRAY2   (I4/1:10,1:V) /* first dimension fixed and second variable
1 #ARRAY3   (I4/1:V,1:*)  /* first dimension variable and second eXtensible
. . .
END-DEFINE
```

Extracting Natural Structure Information (IDL Levels)

Source Extractions

Natural levels are always kept. This means that the structure in the extracted IDL is the same as in the original Natural program.

Object Extractions

- **UNIX or Windows**
  In UNIX or Windows RPC environments, Natural levels are *not* kept. The IDL is extracted in a flat way, where
  - all IDL parameters are at level 1;
  - all Natural groups are removed;

  - Natural fields within groups using repetition (`PERIODIC GROUPS`) are mapped to IDL arrays;
  - the dimension of Natural arrays within groups using repetion (`PERIODIC GROUPS`) is increased in the IDL. For example, a one-dimensional array may become a two-dimensional or three-dimensional IDL array depending on the dimension of the group;

- **z/OS**
  In z/OS RPC environments, the Natural programs must be compiled (cataloged) with the compiler option `SYMGEN=ON` to keep Natural levels, otherwise flat extraction is carried out.

Extracting Parameters defined with OPTIONAL

For a parameter defined without `OPTIONAL`, a value must be passed from the invoking Natural object, i.e. the caller.

For a parameter defined with `OPTIONAL`, a value can, but need not be passed from the invoking Natural object to this parameter. With the `SPECIFIED` option, a Natural server can find out at runtime whether an optional parameter has been defined or not.

The IDL Extractor for Natural ignores the `OPTIONAL` specification, i.e. the parameter is extracted as without the `OPTIONAL` specification. See the *Natural Documentation* for more information.

EntireX RPC technology does *not* support optional IDL parameters. Using pure Natural RPC (Natural client to Natural server), Natural optional parameters are supported.

Setting Natural Parameters to Constants

Setting parameters to constant values and suppressing them in the IDL is part of the redesign process of the extracted interface. This keeps the IDL client interface lean. See *An Example for Set Constant*.

EntireX and Natural RPC make sure the constant value is passed to the Natural server during runtime. No data is transferred between the RPC client and the RPC server.

For more information, see *Step 6: Redesign the Interface for Natural Subprograms (Optional)*.

Suppressing Natural Parameters

Hiding or suppressing unneeded parameters in the IDL is part of the redesign process of the extracted interface. This keeps the IDL client interface lean and minimizes the amount of data to be transferred during runtime.

EntireX and Natural RPC make sure to provide low values as input for suppressed parameters to the Natural server called (blank for IDL type `A`, zero for numeric data types such as IDL `I`, `N` and `P`). No data is transferred between an RPC client and the RPC server.

For more information, see *Step 6: Redesign the Interface for Natural Subprograms (Optional)*.

Renaming a Program

Renaming a program to a different name in the IDL is part of the redesign process of the extracted interface. You can adjust the short Natural name to a meaningful longer name for better readability. See *An Example for Extracting Multiple Interfaces* where the original Natural name `CALC` is renamed to IDL names `ADD`, `SUBTRACT`, `MULTIPLY` etc.

EntireX and Natural RPC make sure the original Natural server is called during runtime. For more information, see *Step 6: Redesign the Interface for Natural Subprograms (Optional)*.