

Generating Natural Data Definition Modules (DDMs)

To enable Natural to access an SQL table, a logical Natural data definition module (DDM) of the table must be generated. This is done either with Predict (see the relevant Predict documentation for details) or with the Natural utility `SYSDDM`.

If you do not have Predict installed, use the `SYSDDM` function **SQL Services** to generate Natural DDMs from SQL tables. This function is invoked from the main menu of `SYSDDM` and is described on the following pages.

For further information on Natural DDMs, see *Data Definition Modules - DDMs* in the *Natural Programming Guide*.

This section covers the following topics:

- SQL Services (NSB)
-

SQL Services (NSB)

To access SQL tables, you may use the **SQL Services (NSB)** function of the Natural `SYSDDM` utility; see *Function Code Z* in the section *Description of Functions* in the *Natural Editors* documentation. You access the CXX CDD (ConnecX data dictionary) of your current CXX connection to retrieve table definitions for Natural DDM generation. The name of the CDD catalog you access is displayed in the top left-hand corner of the screen SQL Services Menu. You can access any catalog contained in the CDD. For further details on the CDD structure read the *ConnecX* documentation.

To invoke the SQL Services (NSB) function

1. In the command line, enter the Natural system command `SYSDDM`.

The menu of the `SYSDDM` utility appears.

2. In the **Code** field, enter function code `Z`.

A menu is displayed, which offers you the following functions:

- Select Catalog name from a List
- CXX Connection handling
- Select SQL Table from a List
- Generate DDM from an SQL Table
- List Columns of an SQL Table

These Functions are described in the following sections.

Select Catalog Name from a List

This function is used to select a catalog from the catalogs defined in the CDD for further processing.

▶ To invoke the Select Catalog Name from a List function

1. On the **SQL Services (NSB)** menu, enter function code C and press Enter.

A list of all catalogs defined in the current CDD is displayed.

2. On the list, you can mark an SQL catalog with an S to select a catalog for further processing.

The selected catalog is displayed in the left corner of the second header line of following maps and in the **Catalog name** field of the **SQL Services (NSB)** menu, where the catalog name could also be entered. If you did not explicitly specify a catalog name it is set to either the current default catalog of the CXX connection – if it is not equal spaces – or the first catalog found in the current CDD.

CXX Connection Handling

This function is used to verify and to change the actual CXX connection. It displays the current parameters of the connection.

▶ To invoke the CXX Connection Handling function

- On the **SQL Services (NSB)** menu, enter function code X and press Enter.

The **CXX Connection Handling** screen is displayed.

The following parameters are available:

Parameter	Description
GATEWAY	Specifies the IP-address of the CXX server connected to.
DD	Specifies the registered data source name of the CDD in use.
PORT	Specifies the port number the CXX server is listening to.
User	Specifies the user name of the CXX connection.
Password	Specifies the password used for the CXX connection(invisible).
Catalog	Specifies the current default catalog name of the CXX connection.
Schema	Specifies the current default schema name of the CXX connection.
Version	Displays the RCI version string of the CXX connection.
State	Displays the CXX connection state.

▶ To change the parameters of the connection

- Enter the new parameter data and press PF5 (Update).

The connection is (re-)established with the entered parameters.

Select SQL Table from a List

This function is used to select an SQL table from a list for further processing.

To invoke the Select SQL Table from a List function

1. On the **SQL Services (NSB)** menu, enter function code S.
 - If you enter the function code only, you obtain a list of all tables defined in the selected SQL catalog.
 - If you do not want a list of all tables but would like only a certain range of tables to be listed, you can, in addition to the function code, specify a start value in the **Table Name** and/or **Schema** fields. You can also use asterisk notation (*) for the start value.

Press Enter.

The **Select SQL Table from a List** screen appears, displaying a list of all SQL tables requested.

2. On the list, you can mark an SQL table with either G for **Generate DDM from an SQL Table** or L for **List Columns of an SQL Table**.

Press Enter.

The selected function is displayed for the marked table. For further information, see the corresponding descriptions in the following sections.

Generate DDM from an SQL Table

This function is used to generate a Natural DDM from a DB2 table, based on the definitions in the DB2 catalog.

The following topics are covered below:

- Invoking the Generate DDM from an SQL Table function
- DBID/FNR Assignment
- Long Field Redefinition
- Length Indicator for Variable Length Fields: VARCHAR, LONG VARCHAR, VARGRAPHIC, LONG VARGRAPHIC
- Null Values

Invoking the Generate DDM from an SQL Table function

To invoke the Generate DDM from an SQL Table function

- On the **SQL Services (NSB)** menu, enter function code G along with the name and creator of the table for which you wish a DDM to be generated.
 - If you do not know the table name/schema, you can use the function **Select SQL Table** from a list to choose the table you want.

- If you do not want the schema name of the table to be part of the DDM name, enter an N in the field **DDM Name with Creator** (default is Y).

Important:

Since the specification of any special characters as part of a field or DDM name does not comply with Natural naming conventions, any special characters allowed within SQL must be avoided. SQL delimited identifiers must be avoided, too.

- If you wish to generate a DDM for a table for which a DDM already exists and you want the existing one to be replaced by the newly generated one, enter a Y in the **Replace** field.

By default, **Replace** is set to N to prevent an existing DDM from being replaced accidentally.

Note:

If **Replace** is N, you cannot generate another DDM for a table for which a DDM has already been generated.

DBID/FNR Assignment

When the function **Generate DDM from an SQL Table** is invoked for a table for which a DDM is to be generated for the first time, the **DBID/FNR Assignment** screen is displayed.

If a DDM is to be generated for a table for which a DDM already exists, the existing DBID and FNR are used and the **DBID/FNR Assignment** screen is suppressed.

On the **DBID/FNR Assignment** screen, enter one of the database IDs (DBIDs) chosen at Natural installation time, and the file number (FNR) to be assigned to the DB2 table. Natural requires these specifications for identification purposes only.

The range of DBIDs which is reserved for SQL tables is specified in the NTDB parameter macro of the Natural parameter module (see the Natural *Parameter Reference* documentation) in combination with the NDBID macro of the parameter module NDBPARM. Any DBID not within this range is not accepted. The FNR can be any valid file number within the database (between 1 and 255).

Long Field Redefinition

The maximum field length supported by CXX is 32 KB - 1. If an SQL table contains a column which is longer than 253 bytes, the pop-up window **Long Field Generation** will appear automatically.

A field which is longer than 253 bytes may be defined as a simple Natural field with a maximum length of 32 KB -1, or as an array. In the DDM, such an array is represented as a multiple-value variable.

If, for example, a DB2 column has a length of 2000 bytes, you can specify an array element length of 200 bytes, and you receive a multiple-value field with 10 occurrences, each occurrence with a length of 200 bytes.

Since redefined long fields are not multiple-value fields in the sense of Natural, the Natural C* notation makes no sense here and is therefore not supported.

When such a redefined long field is defined in a Natural view to be referenced by Natural SQL statements (that is, by host variables which represent multiple-value fields), both when defined and when referenced, the specified range of occurrences (index range) must always start with occurrence 1. If not, a Natural syntax error is returned.

Example:

```
UPDATE table SET varchar = #arr(*)
SELECT ... INTO #arr(1:5)
```

Note:

When such a redefined long field is updated with the Natural DML UPDATE statement (see the relevant section in the *Statements* documentation), care must be taken to update each occurrence appropriately.

Length Indicator for Variable Length Fields: VARCHAR, LONG VARCHAR, VARGRAPHIC, LONG VARGRAPHIC

For each of the columns listed above, an additional length indicator field (format/length I2) is generated in the DDM. The length is always measured in number of characters, not in bytes. To obtain the number of bytes of a VARGRAPHIC or LONG VARGRAPHIC field, the length must be multiplied by 2.

The name of a length indicator field begins with L@ followed by the name of the corresponding field. The value of the length indicator field can be checked or updated by a Natural program.

If the length indicator field is not part of the Natural view and if the corresponding field is a redefined long field, the length of this field with UPDATE and STORE operations is calculated without trailing blanks.

Null Values

With Natural, it is possible to distinguish between a null value and the actual value zero (0) or blank in an SQL column.

When a Natural DDM is generated from the SQL catalog, an additional NULL indicator field is generated for each column which can be NULL; that is, which has neither NOT NULL nor NOT NULL WITH DEFAULT specified.

The name of the NULL indicator field begins with N@ followed by the name of the corresponding field.

When the column is read from the database, the corresponding indicator field contains either zero (0) (if the column contains a value, including the value 0 or blank) or -1 (if the column contains no value).

Example:

The column NULLCOL CHAR(6) in an SQL table definition would result in the following view fields:

```
NULLCOL A 6.0
N@NULLCOL I 2.0
```

When the field NULLCOL is read from the database, the additional field N@NULLCOL contains:

- 0 (zero) if NULLCOL contains a value (including the value 0 or blank),
- -1 (minus one) if NULLCOL contains no value.

A null value can be stored in a database field by entering -1 as input for the corresponding NULL indicator field.

Note:

If a column is NULL, an implicit RESET is performed on the corresponding Natural field.

List Columns of an SQL Table

This function lists all columns of a specific SQL table.

 **To invoke the List Columns of an SQL Table function**

- On the **SQL Services Menu**, enter Function Code L along with the name and creator of the table whose columns you wish to be listed, and press Enter.

The **List Columns** screen for this table is invoked, which lists all columns of the specified table and displays the following information for each column:

Variable	Content
Name	The name of the column.
Type	The column type.
Length	The length (or precision if Type is DECIMAL) of the column as defined in the DB2 catalog.
Scale	The decimal scale of the column (only applicable if Type is DECIMAL).
Update	Y - The column can be updated. N - The column cannot be updated.
Nulls	Y - The column can contain null values. N - The column cannot contain null values.
Not	A column which is of a scale length or type not supported by Natural is marked with an asterisk (*). For such a column, a view field cannot be generated. The maximum scale length supported is 7 bytes. Types supported are: CHAR, VARCHAR, LONG VARCHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DECIMAL, INTEGER, SMALLINT, DATE, TIME, TIMESTAMP, FLOAT, ROWID, BLOB, CLOB and DBCLOB.

The data types DATE, TIME, TIMESTAMP, FLOAT and ROWID are converted into numeric or alphanumeric fields of various lengths: DATE is converted into A10, TIME into A8, TIMESTAMP into A26, FLOAT into F8 and ROWID into A40.

For SQL, Natural provides an SQL TIMESTAMP column as an alphanumeric field (A26) in the format *YYYY-MM-DD-HH.SS.MMMMMM*.

Since Natural does not yet support computations with such fields, a Natural subprogram called NDBSTMP is provided to enable this kind of functionality.