

Database Access with Natural for Tamino

This section describes the different aspects of accessing a Tamino database with the Natural data manipulation language (DML).

Tamino stores structured data-oriented XML documents in containers called doctypes. The doctypes are grouped logically together in so-called collections. Collections are stored in a Tamino database, which is the physical container of data. The kind of data that can be stored in Tamino and that is to be accessed by Natural for Tamino must be defined in a Tamino XML Schema.

For information about how to configure Natural to work with Tamino, see the section Natural and Tamino Access in the Natural for Windows documentation.

DDM and view-definitions with Natural for Tamino

This section describes the composition of DDMs and views, as well as their connection to the Tamino XML Schema definition.

DDMs for Natural for Tamino

For Natural to be able to access a Tamino database, a logical connection between a Tamino doctype and the Natural data structures must be provided. Such a logical connection is called a DDM (Data Definition Module). A DDM is the Natural representation of a doctype in a Tamino database. The DDM contains information about the type of each data field and all the necessary structural information as defined in the corresponding Tamino XML Schema.

You define DDMs with the Natural DDM Editor. For more information about Tamino XML Schema mapping, see the section DDM Editor in the Natural for Windows documentation.

Graphik

Each DDM field holds the following information:

Column	Explanation						
T	Field type: <table border="1" data-bbox="363 1487 1439 1702"> <tbody> <tr> <td>blank (no entry)</td> <td>Elementary field. This type of field can hold data and does not contain any other fields.</td> </tr> <tr> <td>G</td> <td>Group. A group is a collection of fields defined under one common group name. Such fields cannot hold any data, but are only containers for other fields.</td> </tr> <tr> <td>*</td> <td>Comment line.</td> </tr> </tbody> </table>	blank (no entry)	Elementary field. This type of field can hold data and does not contain any other fields.	G	Group. A group is a collection of fields defined under one common group name. Such fields cannot hold any data, but are only containers for other fields.	*	Comment line.
blank (no entry)	Elementary field. This type of field can hold data and does not contain any other fields.						
G	Group. A group is a collection of fields defined under one common group name. Such fields cannot hold any data, but are only containers for other fields.						
*	Comment line.						
L	The level number assigned to the field. Levels are used to indicate the structure and grouping of the field definitions. This is relevant with view definitions, redefinitions and field groups.						
Name	The 3 to 32 character external field name. This is the field name used in a Natural program to reference the field. The field name is unique across the whole DDM. It is not necessarily the same name as the TagName.						
F	The format of the field (A=alphanumeric, P=packed numeric, L=logical etc.). Only elementary fields have a format.						

Column	Explanation				
Len	The length of the field. For numeric fields, length is specified as "nn.m", where "nn" is the number of digits before the decimal point and "m" is the number of digits after the decimal point. Alphanumeric fields can have a length entry of DYNAMIC. Only elementary fields can have a length.				
D	<p>The descriptor type of the field:</p> <table border="1" data-bbox="365 546 815 645"> <tr> <td data-bbox="365 546 555 593">D</td> <td data-bbox="555 546 815 593">Elementary descriptor.</td> </tr> <tr> <td data-bbox="365 593 555 645">blank (no entry)</td> <td data-bbox="555 593 815 645">No descriptor.</td> </tr> </table> <p>Descriptors are needed for special clauses (search, sort etc.) in the READ or FIND statements. Only elementary fields, which are not arrays, can be descriptors. For a Tamino XML Schema, an element is marked as a descriptor in the DDM when it has an overall multiplicity of a maximum of 1, in other words, if the maxOccurs values of the element and all of its predecessors are never greater than 1.</p>	D	Elementary descriptor.	blank (no entry)	No descriptor.
D	Elementary descriptor.				
blank (no entry)	No descriptor.				
Header	Indicates a default column header to appear above the field when the field is output via a DISPLAY statement. If no header is specified, the field name is used as column header.				
Edit Mask	Indicates a default edit mask to be used when the field is output via a DISPLAY statement.				
Remarks	This column can contain comments about the field.				
TagName	The name of the tag of the corresponding element in an XML document. This name is defined inside the Tamino XML Schema. This name may be not unique within the whole XML document. Some group fields might not have a TagName.				
Xpath	The complete XPATH in the XML document (as defined in the Tamino XML Schema) for this field. Some group fields might not have an XPATH.				
Occurrence	The multiplicity of the field as extracted from the Tamino XML Schema. The multiplicity of a field is expressed with the maxOccurs facet in the Tamino XML Schema.				

Column	Explanation																								
Flags	<p>Flag field containing additional information for this field as extracted for the Tamino XML Schema. The following flags might appear.</p> <table border="1" data-bbox="363 434 1441 1218"> <tbody> <tr> <td data-bbox="363 434 703 517">ARRAY</td> <td data-bbox="703 434 1441 517">field is an array; i.e. maxOccurs of the corresponding Tamino XML Schema definition is greater than 1</td> </tr> <tr> <td data-bbox="363 517 703 600">GROUP_ATTRIBUTES</td> <td data-bbox="703 517 1441 600">field is a group that contains the attribute sub-fields of the predecessor field</td> </tr> <tr> <td data-bbox="363 600 703 683">GROUP_ALTERNATIVES</td> <td data-bbox="703 600 1441 683">field is a group that represents the choice constructor of Tamino XML Schema; the choice elements are contained as sub-fields</td> </tr> <tr> <td data-bbox="363 683 703 766">GROUP_SEQUENCE</td> <td data-bbox="703 683 1441 766">field is a group that represents the sequence constructor of Tamino XML Schema; the sequence elements are contained as sub-fields</td> </tr> <tr> <td data-bbox="363 766 703 848">GROUP_ALL</td> <td data-bbox="703 766 1441 848">field is a group that represents the all constructor of Tamino XML Schema; all elements are contained as sub-fields</td> </tr> <tr> <td data-bbox="363 848 703 891">ATTR_REQUIRED</td> <td data-bbox="703 848 1441 891">field is an attribute marked as required</td> </tr> <tr> <td data-bbox="363 891 703 934">ATTR_OPTIONAL</td> <td data-bbox="703 891 1441 934">field is an attribute marked as optional</td> </tr> <tr> <td data-bbox="363 934 703 976">ATTR_PROHIBITED</td> <td data-bbox="703 934 1441 976">field is an attribute marked as prohibited</td> </tr> <tr> <td data-bbox="363 976 703 1019">MULT_OPTIONAL</td> <td data-bbox="703 976 1441 1019">field can occur in the XML document but does not need to</td> </tr> <tr> <td data-bbox="363 1019 703 1061">MULT_REQUIRED</td> <td data-bbox="703 1019 1441 1061">field must occur in the XML document</td> </tr> <tr> <td data-bbox="363 1061 703 1104">MULT_ONCE</td> <td data-bbox="703 1061 1441 1104">field must occur exactly once in the XML document</td> </tr> <tr> <td data-bbox="363 1104 703 1218">SIMPLE_CONTENT</td> <td data-bbox="703 1104 1441 1218">field was defined as complexType with simpleContent in the Tamino XML Schema</td> </tr> </tbody> </table> <p>Combinations of the flags for one DDM field are possible.</p>	ARRAY	field is an array; i.e. maxOccurs of the corresponding Tamino XML Schema definition is greater than 1	GROUP_ATTRIBUTES	field is a group that contains the attribute sub-fields of the predecessor field	GROUP_ALTERNATIVES	field is a group that represents the choice constructor of Tamino XML Schema; the choice elements are contained as sub-fields	GROUP_SEQUENCE	field is a group that represents the sequence constructor of Tamino XML Schema; the sequence elements are contained as sub-fields	GROUP_ALL	field is a group that represents the all constructor of Tamino XML Schema; all elements are contained as sub-fields	ATTR_REQUIRED	field is an attribute marked as required	ATTR_OPTIONAL	field is an attribute marked as optional	ATTR_PROHIBITED	field is an attribute marked as prohibited	MULT_OPTIONAL	field can occur in the XML document but does not need to	MULT_REQUIRED	field must occur in the XML document	MULT_ONCE	field must occur exactly once in the XML document	SIMPLE_CONTENT	field was defined as complexType with simpleContent in the Tamino XML Schema
ARRAY	field is an array; i.e. maxOccurs of the corresponding Tamino XML Schema definition is greater than 1																								
GROUP_ATTRIBUTES	field is a group that contains the attribute sub-fields of the predecessor field																								
GROUP_ALTERNATIVES	field is a group that represents the choice constructor of Tamino XML Schema; the choice elements are contained as sub-fields																								
GROUP_SEQUENCE	field is a group that represents the sequence constructor of Tamino XML Schema; the sequence elements are contained as sub-fields																								
GROUP_ALL	field is a group that represents the all constructor of Tamino XML Schema; all elements are contained as sub-fields																								
ATTR_REQUIRED	field is an attribute marked as required																								
ATTR_OPTIONAL	field is an attribute marked as optional																								
ATTR_PROHIBITED	field is an attribute marked as prohibited																								
MULT_OPTIONAL	field can occur in the XML document but does not need to																								
MULT_REQUIRED	field must occur in the XML document																								
MULT_ONCE	field must occur exactly once in the XML document																								
SIMPLE_CONTENT	field was defined as complexType with simpleContent in the Tamino XML Schema																								
DefaultValue	default value of the field as extracted from Tamino XML Schema; this is not yet used																								
FixedValue	fixed value of the field as extracted from Tamino XML Schema; this is not yet used																								

Note:

Some of the field information data is read-only and cannot be changed by the user.

The following values can be changed by the user:

- Name
- Format
- Length
- Header
- EditMask

The following values can be adapted when defining a view from a DDM:

- Name
- Format
- Length
- Occurrence

XPATH Information in a DDM Field

The XPATH information stored in a Natural for Tamino DDM is used during application runtime to uniquely identify a data element in a given XML document. Therefore, it is not possible to change the XPATH information.

Flags in a DDM Field

The flags of a field in a Natural DDM are used internally to help in correctly recognizing special group structures (i.e. the attributes of an element tag) or multiple occurrences. Additionally, the user can identify DDM fields which are either mandatory or optional in XML documents.

Arrays in Natural for Tamino DDMs

If you define an XML element with a maxOccurs value greater than one in the Tamino XML Schema, then the maxOccurs value overrides the value defined in the Schema. Such a construction is mapped either on a Natural static array definition or on a Natural X-Array definition. Depending on the type of the XML element you are dealing with, the following cases can arise:

- If the XML element is a complexType with complexContent (i.e. it is an element containing other elements) then the generated corresponding Natural group will be an indexed group.
- If the XML element is a simpleType (i.e. the element is holding data only) or a complexType with simpleContent (i.e. the element has only data and attributes but no other elements) then the generated Natural data field will be an array.

For further information about mapping maxOccurs definitions onto Natural arrays, see the section Data Conversion for Tamino. The array boundaries or the kind of the array (static array or X-Array) can be adapted as required in a corresponding view definition.

Example

This is an example of an EMPLOYEES DDM generated from a Tamino XML Schema definition.

The schema can, for example, be defined with the Natural demo application SYSEXINS:

```

DB: 00250 FILE: 00001 - EMPLOYEES-XML
TYPE: XML
COLLECTION: NATDemoData
SCHEMA: Employee
DOCTYPE: Employee
NAMESPACE-PREFIX: xs
NAMESPACE-URI: http://www.w3.org/2001/XMLSchema
T L Name F Leng D Remark
-----
G 1 EMPLOYEE
  FLAGS=MULT_REQUIRED,MULT_ONCE
  TAG=Employee
  XPATH=/Employee
G 2 GROUP$1
  FLAGS=GROUP_ATTRIBUTES
  3 PERSONNEL-ID A 8 D xs:string
  FLAGS=ATTR_REQUIRED
  TAG=@Personnel-ID
  XPATH=/Employee/@Personnel-ID
G 2 GROUP$2
  FLAGS=GROUP_SEQUENCE,MULT_REQUIRED,MULT_ONCE
G 3 FULL-NAME
  FLAGS=MULT_OPTIONAL
  TAG=Full-Name
  XPATH=/Employee/Full-Name
G 4 GROUP$3
  FLAGS=GROUP_SEQUENCE,MULT_REQUIRED,MULT_ONCE
  5 FIRST-NAME A 20 D xs:string
  FLAGS=MULT_OPTIONAL
  TAG=First-Name
  XPATH=/Employee/Full-Name/First-Name
  5 MIDDLE-NAME A 20 D xs:string
  FLAGS=MULT_OPTIONAL
  TAG=Middle-Name
  XPATH=/Employee/Full-Name/Middle-Name
  5 MIDDLE-I A 20 D xs:string
  FLAGS=MULT_OPTIONAL
  TAG=Middle-I
  XPATH=/Employee/Full-Name/Middle-I
  5 NAME A 20 D xs:string
  FLAGS=MULT_OPTIONAL
  TAG=Name
  XPATH=/Employee/Full-Name/Name
  . . .
  3 LANG A 3 xs:string
  FLAGS=ARRAY,MULT_OPTIONAL
  OCC=1:4
  TAG=Lang
  XPATH=/Employee/Lang

```

Definition of VIEWS

In order to work with Tamino database fields in a Natural program, you must specify the required fields of the DDM in a Natural *view-definition* (see the DEFINE DATA statement). Normally, a view is a special subset of the complete data structure as defined in the DDM.

Tamino XML Schema->Natural for Tamino DDM->Natural *view-definition*

A view for the EMPLOYESS-XML DDM, where one of the view fields is a static array, might look like this:

```
DEFINE DATA LOCAL
01 VW VIEW OF EMPLOYEES-XML
02 NAME
02 CITY
02 LANG (1:4)
END-DEFINE
```

Natural Statements for Tamino Database Access

The Natural DML statements which are provided for Tamino access can be subdivided into two categories:

- pure retrieval statements;
- database modification statements.

The Natural system variable *ISN is mapped on the Tamino ino:id.

Natural for Tamino Retrieval Statements

The following Natural statements can be used for database retrieval:

- FIND
This statement is used to select those records from a database which meet a specified search criterion.
- GET
This statement is used to select one special record with its unique id from the database.
- READ
This statement is used to select a range of records from a database in a specified sequence.

Not all of the possible options and all of the possible clauses of the retrieval statements can be used for Tamino access. Please read the appropriate section in the Statements documentation for a detailed description.

All are internally realized with the Tamino _xquery command verb. Statement clauses are mapped to corresponding Tamino XQuery expressions, e.g. search criteria are mapped to Tamino XQuery comparison expressions, sequence specifications are mapped to Tamino XQuery ordering expressions with sort direction.

The result set for the FIND and READ statements is determined at start of the loop and remains unchanged throughout the loop.

The following is an example of reading a set of employee records from a Tamino database where one view field is an array:

```
* READ 5 RECORDS DESCENDING CONTAINING A
* STATIC ARRAY IN THE VIEW DEFINE DATA LOCAL
01 VW VIEW OF EMPLOYEES-TAMINO
02 NAME
02 CITY
02 LANG (1:4)
END-DEFINE
*
READ(5) VW DESCENDING BY NAME = 'MAYER'
  DISPLAY NAME CITY LANG(*)
END-READ
*
END
```

Natural for Tamino Database Modification Statements

The following database modification statements are provided for use with Natural for Tamino:

- **STORE**
This statement is used for inserting a new XML document into the database.
- **DELETE**
This statement is used for deleting a document from the database. The DELETE statement implements a positioned delete.

For a detailed description of the statements, see the appropriate sections of the Natural Statements documentation.

The DELETE statement is internally realized with the Tamino `_delete` command verb using the current `ino:id`, and the STORE statement is implemented with the `_process` command verb.

The following example program stores a new employee record with some data in the database:

```

* STORE NEW EMPLOYEE
DEFINE DATA LOCAL
01 VW VIEW OF EMPLOYEES-TAMINO
02 PERSONNEL-ID
02 NAME
02 CITY
02 LANG (1:3)
END-DEFINE
*
* FILL VIEW
PERSONNEL-ID := '1230815'
NAME          := 'KENT'
CITY          := 'ROME'
LANG(1)       := 'ENG'
LANG(2)       := 'GER'
LANG(3)       := 'SPA'
*
* STORE VIEW
STORE RECORD IN VW
*
COMMIT
*
END

```

If the Tamino XML Schema defines data structures for a doctype as being mandatory, then these data structures must also be filled in the view before a STORE statement is issued, otherwise this will result in a Tamino error.

Natural for Tamino Logical Transaction Handling

Natural performs database modification operations based on transactions, which means that all database modification requests are processed in logical transaction units. A logical transaction is the smallest unit of work (as defined by you) which must be performed in its entirety to ensure that the information contained in the database is logically consistent.

A logical transaction may consist of one or more modification statements (DELETE, STORE) involving one or more doctypes in the database. A logical transaction may also span multiple Natural programs.

A logical transaction begins when a database modification statement is issued. Natural does this automatically. For example, if a FIND loop contains a DELETE statement. The end of a logical transaction is determined by an END TRANSACTION statement in the program. This statement ensures that all modifications within the transaction have been successfully applied.

Natural for Tamino Error Handling

In addition to Natural's standard error messages there are two special error codes which provide additional information via a sub-error code.

- NAT8400 "Tamino error occurred" :
For this special error an additional sub-code number is shown. This number refers to a Tamino error message. Please see the Tamino Messages and Codes documentation.
- NAT8411 "HTTP request failed with response code" : The error code from the http server is delivered as additional information. Please see the REQUEST DOCUMENT documentation, section operand13, for an overview of the response codes for HTTP/HTTPs requests.

Example of Natural for Tamino Interacting with a SQL Database

This is a more sophisticated example of Natural for Tamino interacting with an SQL database; it retrieves data from a Tamino database and inserts or updates the corresponding row in an appropriate table in a SQL database.

```

*
* TAMINO DB --> SQL RDBMS EXAMPLE
*
DEFINE DATA LOCAL
* DEFINE VIEW FOR TAMINO
01 VW-TAMINO VIEW OF EMPLOYEES-TAMINO
02 PERSONNEL-ID
02 NAME
02 CITY
* DEFINE VIEW FOR SQL DATABASE
01 VW-SQL VIEW OF EMPLOYEES-SQL
02 PERSONNEL_ID
02 NAME
02 CITY
END-DEFINE
*
* OPEN A TAMINO LOGICAL READ LOOP
*
TAMINO. READ VW-TAMINO BY NAME
*
* SEARCH RECORD IN SQL DATABASE AND
* INSERT A NEW RECORD IF NOT FOUND OR
* UPDATE THE EXISTING ONE WITH THE DATA
* FROM TAMINO DB
*SQL. FIND(1) VW-SQL WITH PERSONNEL_ID = PERSONNEL-ID (TAMINO.)
      IF NO RECORDS FOUND
          PERSONNEL_ID := PERSONNEL-ID (TAMINO.)
          NAME           := NAME           (TAMINO.)
          CITY           := CITY           (TAMINO.)
          STORE VW-SQL
          ESCAPE BOTTOM
      END-NOREC
      PERSONNEL_ID := PERSONNEL-ID (TAMINO.)
      NAME           := NAME           (TAMINO.)
      CITY           := CITY           (TAMINO.)
      UPDATE
      END-FIND
*
END-READ
*
END TRANSACTION
*
END

```

Natural for Tamino Restrictions

There are restrictions concerning the scope of the Tamino XML Schema language that can be used for creating schemas for Natural for Tamino DDMs generation:

- Only Tamino XML Schema language constructors and attributes (as mentioned in the DDM Editor for Windows documentation) are supported by Natural for Tamino. Other constructors like `xs:any`, `xs:anyAttribute`, etc. cannot be applied in Tamino XML Schemas if you wish to use them together with Natural for Tamino.
- The functionality of `xs:import` is not supported by Natural for Tamino. This means that external schema components must not be referenced in a Tamino XML Schema suitable for usage together with Natural. In other words, a doctype definition in a Tamino XML Schema must resolve all references within this Tamino XML Schema itself if you are planning to use it together with Natural for Tamino.
- The attribute `mixed` of the constructor `xs:complexType` is only supported with its default value "false". Natural for Tamino does not support mixed-content document definitions (as set with the specification `mixed="true"`). Using `mixed="true"` will result in an error during DDM generation.

- The level of nested structures in a Natural for Tamino DDM is limited to 99. A new DDM level is generated whenever one of the following constructors occurs in the Tamino XML Schema: `xs:element`, `xs:attribute`, `xs:choice`, `xs:all`, `xs:sequence`.
- Recursively defined structures in a Tamino XML Schema cannot be used together with Natural for Tamino.
- The Tamino XML Schema language constructor `xs:choice` is mapped on a Natural group containing all alternatives of the choice. To restrict processing to one particular choice, an appropriate view with the required choice has to be created.