Migration Strategies for the Manual Migration Process

This section describes the possible strategies for migrating Tamino schemas manually under the following headings:

- Comparison to Former Tamino Schema Language
- Migrating a Schema from a Previous Version of Tamino
- Defining a Schema from a DTD
- Defining a Schema from a given XML Schema
- Defining a Schema from Scratch

Comparison to Former Tamino Schema Language

In the former versions of Tamino, the *schema* language is implemented as an XML *Document Type Definition* (*DTD*). This DTD consists of three elements, "*Collection*", "*Doctype*" and "*Node*", which stand in the following relation to each other:

```
<!ELEMENT
Collection (Doctype*)> <!ELEMENT Doctype (Node+)> <!ELEMENT Node
EMPTY>
```

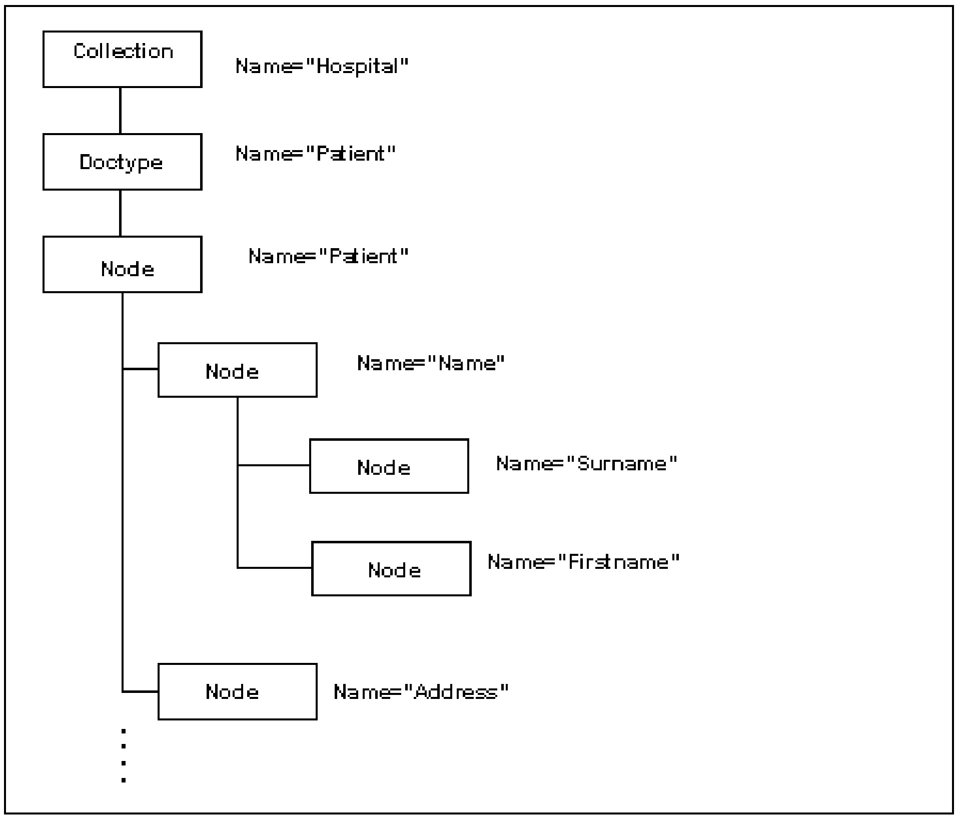
You define a schema by specifying values for the attributes on these elements. The attributes are not shown here: for an overview of the Tamino schema attributes, see Quick Reference to the Mapping Language.

- A Collection relates to a database. The Collection element therefore has no parent element, though the Data Map can contain multiple Collections. A Collection can contain zero or more Doctype elements. Collections are used to combine Doctypes for the purpose of providing parallel access to multiple Doctypes. Examples of Collections are: a hospital’s administration database; an inventory database of a supplier of spare parts; an on-line library.

- A Doctype is a data definition, comparable to Tables or Views used in relational database systems. Doctypes relate to "real-world" data objects such as a patient’s health record in a hospital database, a particular spare part in a spare parts database, or a book in an online library. One Tamino schema describes one Doctype, so that the two terms are sometimes used synonymously (for example, an "instance of the schema" could also be said to be "an instance of the Doctype"). A Doctype must contain one or more Node elements.

- The Node element is empty. Nodes express all the information items contained in the Doctype, for example, a patient’s name, the article number of a spare part, the title of a book. Nodes can be arranged in a tree structure by using attributes to express parent-child relationships. Some Nodes are thus intermediate Nodes, describing a path to the actual information item. For example, since a patient’s name can be further structured into a first name and surname, the Node representing the patient’s name is defined as an intermediate Node, whereas the Nodes representing first name and surname contain character data and are therefore referred to as "terminal Nodes" or "leaves".

Using the example of a hospital database, the following figure illustrates the schema of a patient’s record (Doctype "patient"):



The figure shows that Doctype structures can be seen as trees. Intermediate Nodes and their subordinate Nodes are "branches" or "subtrees". Terminal Nodes (or "leaves") are those that contain "real" data. In the above example, the Node with Name="Name" is an intermediate Node, the Nodes "Surname" and "Firstname" are terminal Nodes. A Collection, as a collection of trees, is said to be a "grove".

The figure also gives you some idea of how, by specifying appropriate attributes, subtrees or Nodes can be assigned to be stored in different data locations.

The Name attribute is available for naming the Collection, the Doctype and the Nodes. Note that the name of the Doctype must be same as the name of the root (top-level) Node. Other essential attributes are ID-declared attributes used to uniquely identify information items, and IDREF-declared attributes to representing the relationship between information items. Other attributes relate to parameters relevant to a specific database (for example, SQL or Adabas-specific attributes) and whether or how data is to be indexed.

Migrating a Schema from a Previous Version of Tamino

The current version of Tamino allows you to migrate schemas defined using versions of Tamino up to 2.2.

These earlier versions used schemas based on the Tamino Data Map DTD and can be migrated to the Tamino schema language that is based on the *XML Schema* standard and used in versions of Tamino higher than 2.2.

The easiest way of migrating an "old" schema to the *XML Schema*-based Tamino Schema Language is to read the schema into the Schema Editor 3 and redefining the schema to Tamino. The Schema Editor has a built-in schema converter that will express the schema in *XML schema* syntax. You can, of course, refine the schema using the supported XML Schema features.

For more information on migrating schemas, see the next section of this document, ’Migration from Old to New Schema Language’.

Note:
Once you have migrated a schema, you must also reload (reindex) its document instances. The current version of Tamino supports the "old" schema language, thus allowing you to migrate schemas and instances over a period of time. You are, however, strongly recommended to proceed with migration, as support for the "old" syntax will be phased out.

Defining a Schema from a DTD

Tamino allows you to start the schema definition process using an arbitrary external DTD.

Using the Import DTD function of the Schema Editor 3, you can read a DTD into the editor and either use default mapping for the elements and attributes of the DTD, or you can specifically map each element/attribute that interests you.

A combination of both is also possible: you define default mapping, and refine the mapping information for appropriate nodes, using the features of XML Schema that cannot be inferred from DTDs. Key XML Schema features not provided for by DTDs are:

- XML Schemas are themselves XML documents and can therefore be parsed as such.
- Contents of elements can be specified as being of a specific data type.
- The number of instances of element types within a parent element can be restricted.
- The order in which instances of different element types can occur within a parent element can be fixed.
- Text values in simple type elements can be restricted.
- The number of items in white-space separated lists that amke up atribute values can be restricted.
- Attribute vales can be specified as a union of different white-space separated lists.

Defining a Schema from a given XML Schema

Tamino allows you to start the schema definition process using an arbitrary external XML schema.

You can read an XML Schema into the Schema Editor 3 and add Tamino-specific information to it. Note, however, Tamino currently supports only a subset of XML Schema features.

Defining a Schema from Scratch

You can define a Tamino schema from scratch, that is, you have neither an existing schema to migrate, nor an external DTD, nor an XML Schema as starting point for schema definition.

You can use the Schema Editor 3 to build a Tamino schema. The advantage of using the Tamino Schema Editor over other schema editing tools is that the Schema Editor 3 will not generate any syntax not supported by Tamino.