Tamino

Getting Started

Version 10.1

April 2018

WEBMETHODS

This document applies to Tamino Version 10.1 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: INS-GETTING-STARTED-101-20180413

Table of Contents

Getting Started

This document is intended for users who want to get started with Tamino. It takes you step by step through some typical basic operations of Tamino, so that you will feel comfortable with the product as quickly as possible. Where relevant, it points you to further documents in the Tamino documentation set where you can get more detailed information on a particular topic.

The following topics are covered:

Introduction	An overview of the main features of Tamino.
Basic Concepts	Definitions of the main concepts of Tamino.
Starting Tamino	How to start Tamino
Creating a Database	Description of how to create a Tamino database.
Starting and Stopping the Database	Information about two basic tasks with databases: How to start and how to stop them.
Working with DTDs and Schemas	Information about creating a Tamino schema from an existing DTD.
Defining a Schema	Description about how to define a schema to a database.
Loading XML Objects into the Database	Information about loading example XML objects into a newly created database.
Loading non-XML Objects into the Database	Description of how to load example non-XML objects into a newly created database.
Retrieving Objects from the Database Using XQuery	Using XQuery to retrieve and manipulate information from an example database.
Backup and Restore	Basic concepts of backup and restore operations for a database.
Where to go from here	Where to find further, in-depth information about Tamino.

1 Introduction

Tamino is a complete database management system for exchanging data and integrating applications on an XML basis. As a native XML database server, it is a powerful technology for making business data available as Internet objects. Tamino optimizes the usage of XML documents, facilitates communication via the Internet, stores XML and non-XML data, and accesses external systems and applications. In this document, the following features of Tamino will be introduced:

- *The Tamino Manager*

  Tamino offers the command line **inoadmin** tool for the administration of Tamino databases. With this tool you can create and delete databases, start and stop databases, define users, perform backup and restore functions, and conduct many other administrative activities.

- *The Tamino Schema Editor*

  The Tamino Schema Editor helps you work with schemas and define them to Tamino. In order to do so, you describe the schema using Tamino's schema definition language, which is based on the XML Schema standard. The Schema Editor allows you to define the schema as a graphical tree and automatically creates the schema definition in Tamino's schema language. Then you load the schema description into the Tamino XML Server. Once a schema has been defined in this way, you can load XML objects that are based on the schema into Tamino.

- *The Tamino X-Plorer*

  The Tamino X-Plorer is a very powerful tool for querying, browsing and manipulating the contents of Tamino databases. It consists of a Content Viewer to display collections, schemas, doctypes and XML or non-XML documents, and a navigation tree to allow easy access to databases and their contents. Also, it provides access to the XQuery Tool for easy data querying using the XQuery language.

This document will take you step by step through the process of creating a database, creating a Tamino schema based on a DTD, defining a schema, loading data into Tamino, and finally querying your data. It is based on a pre-defined example, so that you do not have to bother about creating XML data yourself. We recommend you to read this guide from beginning to end, so that

you will have a complete introduction to the basic functions of Tamino after you have finished reading.

Other tools that are available from the Tamino program group are the Tamino Data Loader, the Tamino Interactive Interface, the XQuery tool and the X-Tension Builder.

2 Basic Concepts: Doctype, Collection, Schema

Before going into the details of database creation and storage and retrieval of XML objects, it is useful to take a look at the three basic concepts used in Tamino: doctype, collection and schema.

Doctype

Generally, a doctype is a root element of a DTD (Document Type Definition), i.e. the outermost element in the document that the DTD applies to. It typically denotes the document's starting and ending point. In Tamino, a doctype represents a container for XML instances with the same root element within a collection.

Collection

The concept of collections is Tamino-specific. In Tamino, a collection is the largest unit of information within a database. Each collection can contain multiple doctypes. It is a container for related information and needs to be defined when defining a Tamino schema.

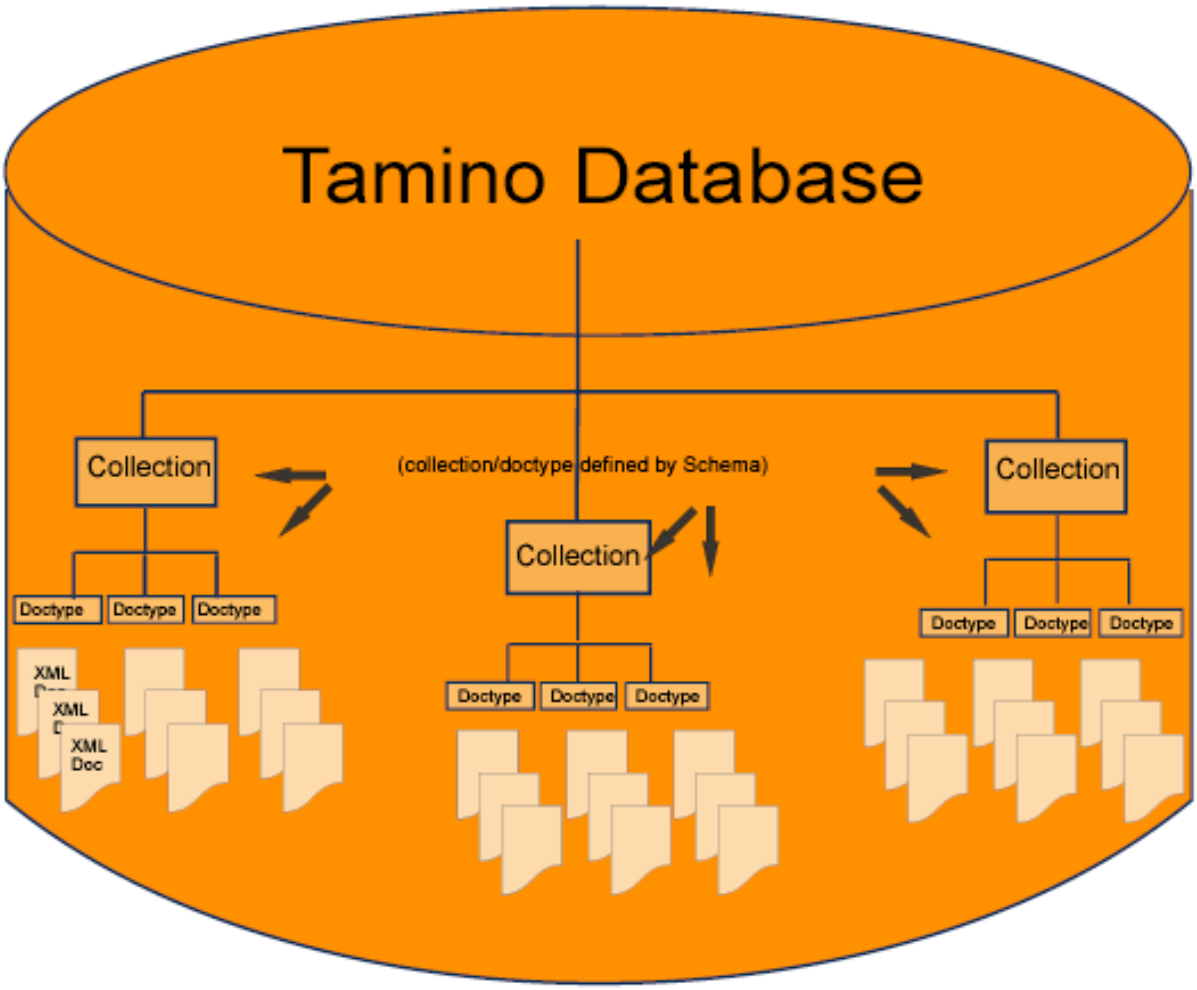
Schema

A Tamino schema complies with a subset of the W3C XML schema standard, with Tamino-specific information defined in annotations. Within a schema, we distinguish between a physical schema and a logical schema.

The logical schema consists of rules describing the relationship between and properties of elements and attributes in valid XML documents.

The physical schema describes how XML documents are physically stored and indexed; changes to the physical schema will not influence the semantics and syntax of Tamino operations.

The following graphic gives an overview over the relationship between basic Tamino concepts:



Basically, the steps that you perform when working with Tamino are:

1. Create a Tamino database and start it.
2. Load objects into the database. This step may require defining a *collection* and a *schema*. If you already have a DTD (or XML Schema) for your XML objects to be loaded, it can be used for *schema definition*.
3. Retrieve your loaded objects by querying the database.
4. Write applications.

In the following, the first three steps will be described in more detail. Information about step four goes beyond the scope of this document. See the documentation about the Tamino APIs and the Advanced Concepts for further information.

3 Starting Tamino

This chapter discusses the following topics:

Starting Tamino on Windows Systems

The individual components of Tamino are available from the Tamino program group under the Windows **Start** menu.

Starting Tamino on UNIX Systems

This section describes the command line procedures that are available for starting the various Tamino components.

The commands (shell scripts) can be found in the directory *$INODIR/$INOVERS/bin*.

```
Command Procedure Name
                   Tamino Component Started

inoadmin.sh
                   Tamino Manager (the Tamino administration
                   application)
inodocu.sh
                   The Tamino user documentation in HTML format
inodocsearch.sh
                   The full text search feature of the Tamino user
                   documentation
inohelp.sh
                   A summary of the available shell scripts
inoint.sh
                   Tamino Interactive Interface (an HTML form for
                   simple interactive communication with Tamino's
                   X-Machine). The Tamino Interactive Interface lets
                   you create and delete collections and schemas, and
                   load and retrieve data.
inoschema.sh
                   Tamino Schema Editor (an application for creating
                   and modifying Tamino schemas)
inosxbuilder.sh
                   Tamino X-Tension Builder (a Java-based application
                   for creating server extensions)

inoload (see note below)
                   Tamino Data Loader (an application for loading and
                   unloading large amounts of data)
inoxplorer.sh
                   Tamino X-Plorer (an application for browsing
```

```
                    Tamino XML Server data)
inoxquery.sh
                    Tamino XQuery Tool (an application for creating,
                    executing and storing Tamino XQuery queries)
```

Notes:

1. *inoload* is an executable file. All of its parameters must be specified directly on the command line.
2. The scripts that start a browser-based client application use the script *inobrowser.sh*, which defines the browser to be used. You might wish to edit the contents of *inobrowser.sh* to point to your preferred browser. By default, the script assumes that you use Mozilla and that the `PATH` environment variable includes the location of the Mozilla executable.
3. The scripts relating to the Tamino user documentation (*inodoc*.sh*) are only available when the user documentation is installed.

Problems with Blocked Content

Software AG documentation uses active content (JavaScript and Java applets). With Service Pack 2 (SP2) for Windows XP, Microsoft introduced a range of powerful new security features. One effect of these security features is that warning messages appear whenever you try to display HTML pages that use active content, for example Software AG documentation, in the Internet Explorer. A typical warning message that appears in the Internet Explorer information bar is:

Caution: To help protect your security, Internet Explorer has restricted this webpage from running scripts or ActiveX controls that could access your computer. Click here for options...

To continue using the documentation, you can do one of the following:

- Use a different web browser. This problem only affects Microsoft's Internet Explorer.

  Note: Software AG does not endorse or recommend any web browser.

- Change the Internet Explorer options to allow active content to run in files on your computer.

To unblock active content

In the Microsoft Internet Explorer:

1 Choose **Tools> Internet Options**.

2 Choose the tab **Advanced**.

3 Scroll down to the section **Security**.

4 Check (tick) the box **Allow active content to run in files on My Computer**.

5 Choose **OK**.

6 Restart the Internet Explorer.

The warning messages should now no longer appear.

- Click on the information bar and choose the option **Allow Blocked Content...**. You will have to do this for each affected page.

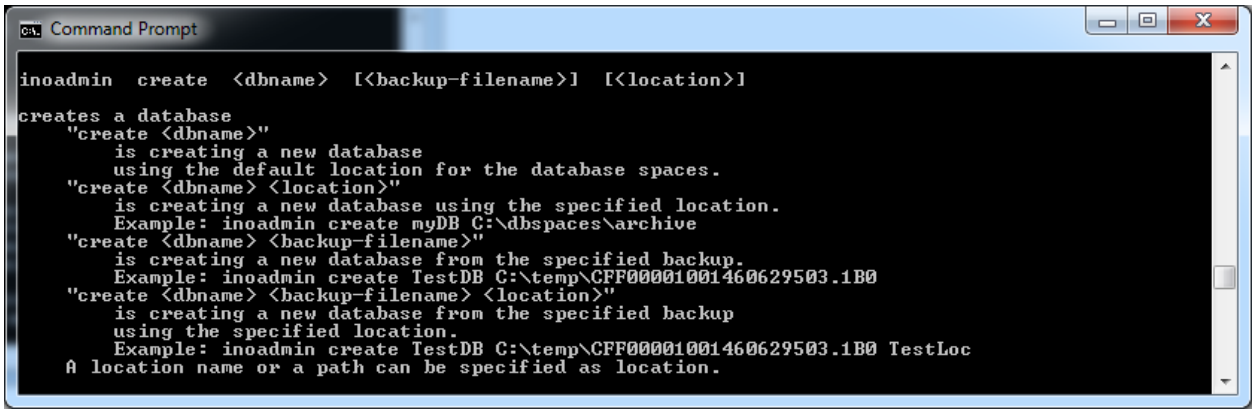
4 Creating a Database

The first thing you need to do when working with Tamino is to create a database. To do so, you use the command line **inoadmin tool**. The command line **inoadmin tool** is a user interface with which you can perform administration functions like starting, stopping, renaming, deleting, and restoring databases. When you create a database, you define a database name, a database location, and database specifics like size and properties. You can either accept the defaults or enter your own preferences according to your specific database needs.

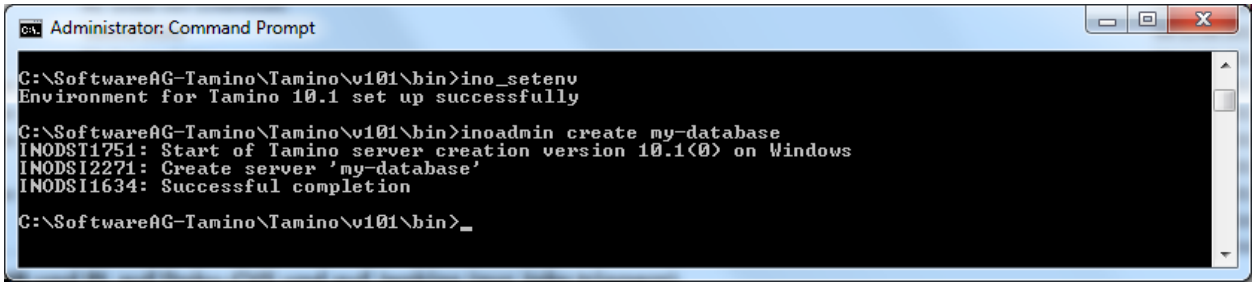
On Windows, the **inoadmin tool** runs in the folder "<INSTALL_ROOT>/Tamino/v101/bin". To set up the administration environment first run the command "ino_setenv".

On UNIX, use the shell script `inoadmin.sh`, which is located in a directory that is included in the search path after the Tamino installation.

Databases are created using **create**.



When only providing the name of the database to-be with the call, database specifics are assigned with default values. The database name can be up to a maximum of 32 characters in length. The following characters may not be used: SPACE , ; ! \ = # . [ ] " % ' / ^ ` ? *

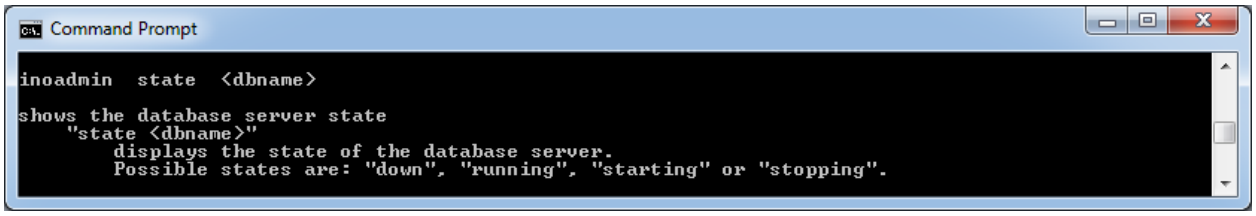


That's all there is to creating a database. If you want to learn more about creating databases, refer to the documentation for the Tamino Manager.

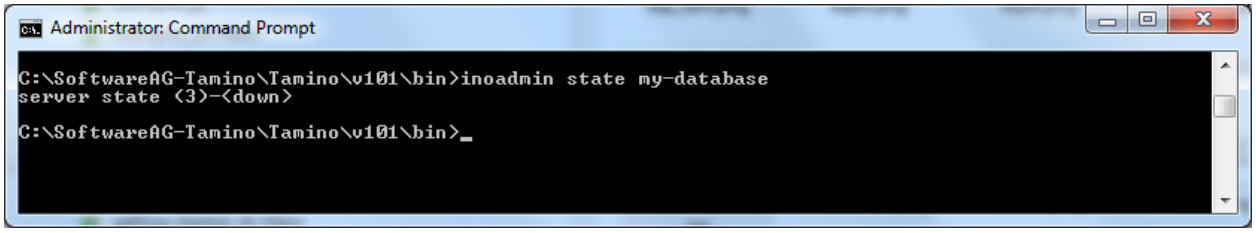
5 Starting and Stopping the Database

The first thing we can do with the database we have just created is to ask its state.

The **state** command shows the state of a dabase server identified by its name.

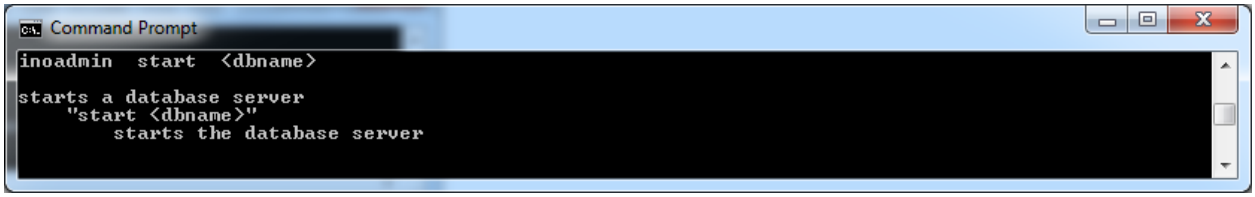


Applying the **state** command upon the database 'my-database' yields 'down', this means that the database is currently stopped.

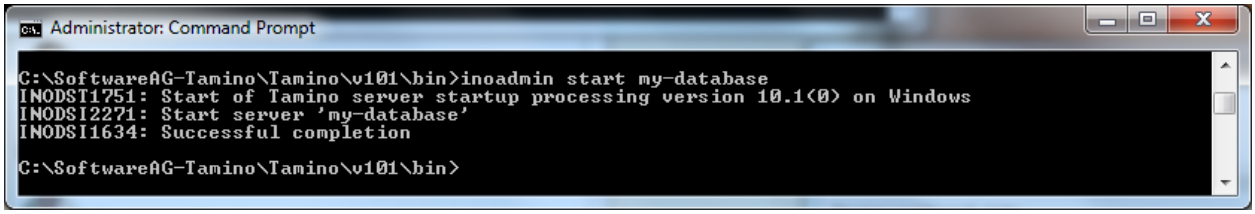


If a database is stopped, you can start it using the inoadmin **start** command.

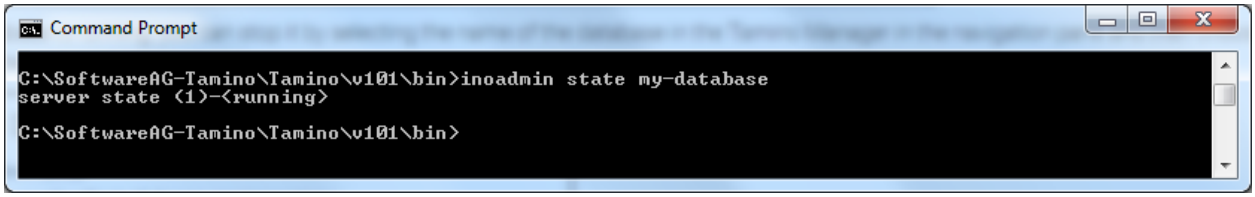
The **start** command starts a dabase server identified by its name.



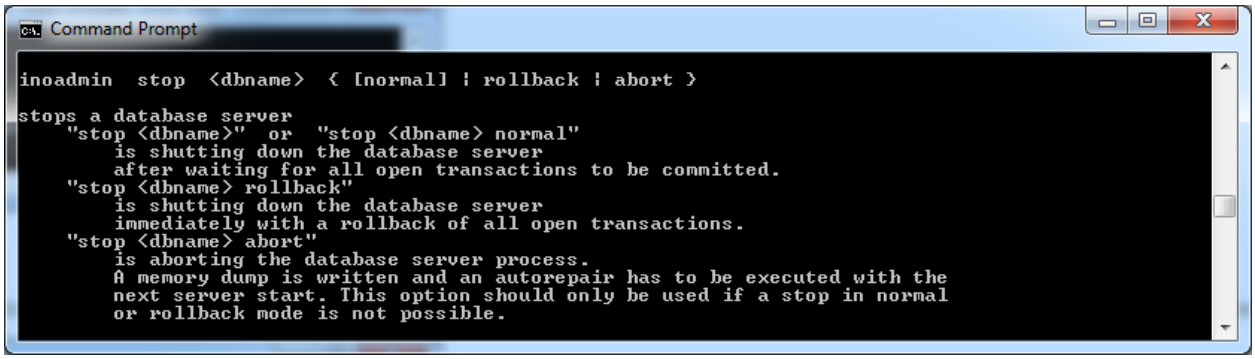
Applying the **start** command upon the database 'my-database' starts the database.



Applying the **state** command upon the database 'my-database' now yields 'running', this means that the database is up and can be worked with.



If a database is running, you can stop it by using the inoadmin **stop** command.



Note: To continue with the following examples, the database needs to be running.

6 Working with DTDs and Schemas

Once your database is up and running, the next step is to work with your XML objects. To do so, you will first need to know how to work with schemas.

A Tamino schema is an XML document that conforms to the W3C's **XML Schema standard**, with Tamino-specific information written in annotations to XML Schema constructs. Experienced Tamino users may well write a schema from scratch, using an editor. Because of the relative complexity of a schema, though, this method is quite error-prone. Another method to write a schema is using Tamino's Schema Editor. The Schema Editor shields you from having to type in schema language syntax, thus making schema creation much faster and less error-prone. A third method, which we will use in the following, is to convert an existing DTD into a Tamino Schema, with the help of Tamino's Schema Editor. This is the easiest method. Then, a schema is generated automatically by Tamino's Schema Editor, based on the sample DTD. The Schema Editor provides default indexing, which can be modified according to your retrieval needs.

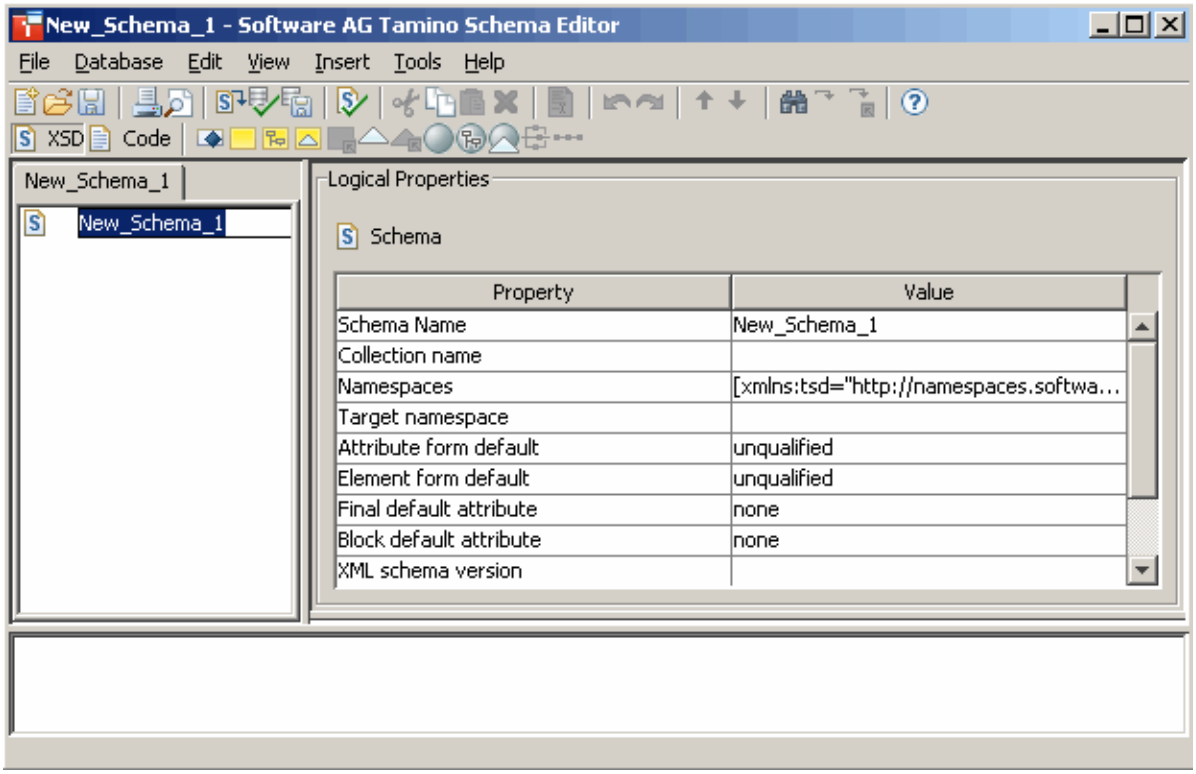
The following example uses the example DTD *patient.dtd* that is supplied with the Tamino product documentation. The examples are located in several subdirectories of `<TaminoDocRootDir>/examples`, where `<TaminoDocRootDir>` is the starting directory of the product documentation. First, we will import the sample DTD into the Tamino Schema Editor.

To import a DTD into the Tamino Schema Editor:

1 On Windows, choose the shortcut **Tamino Schema Editor** in the Tamino program group that is available from the **Start** button.

On UNIX, start *inoschema.sh*.

A blank Tamino Schema Editor appears, showing a schema `New_Schema_1` in the upper left-hand part of the window:

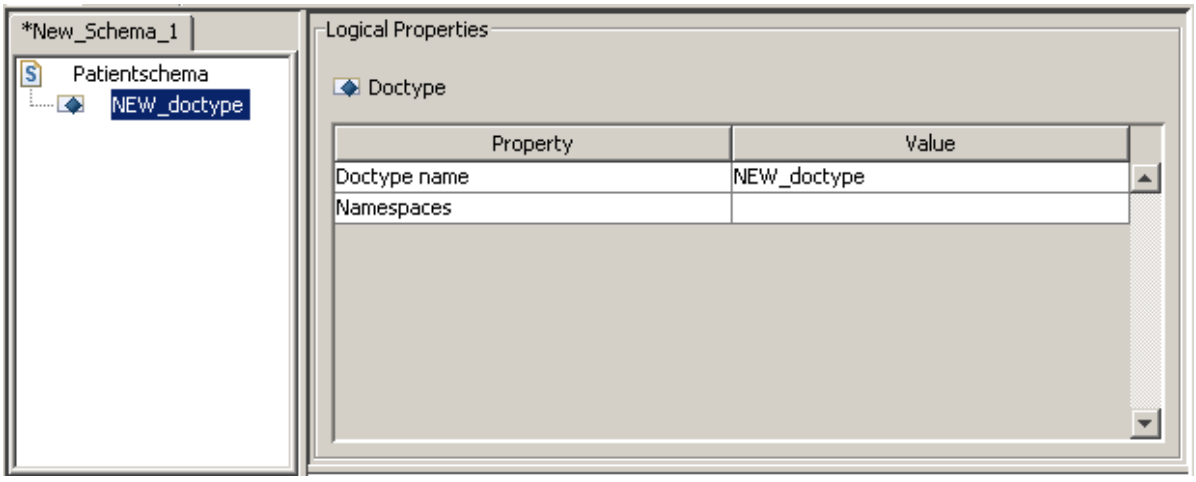


2 Specify a name for your schema to be generated, for example `Patientschema`. To do so, either overwrite the selected text `New_Schema_1`, or select `New_Schema_1` in the **Value** column of the Logical Properties display and type in the new name `Patientschema`.

3 Specify a collection to which the schema is to belong, for example `Hospital`. To do so, place the cursor in the empty field to the right of `Collection name` and type in the new collection name `Hospital`.

4 From the **Insert** menu, choose **Doctype**.

In the tree view, the node `NEW_doctype` appears under `Patientschema`:



5 Name the doctype `patient`. To do so, select `NEW_doctype` in the right-hand part of the window (below **Value**) and enter the new name.

The right-hand part of the editor is called the property sheet. It consists of two parts, the logical properties and the physical properties. The properties are explained in the documentation about the Tamino Schema language.

6 Now we will import a DTD into the schema.

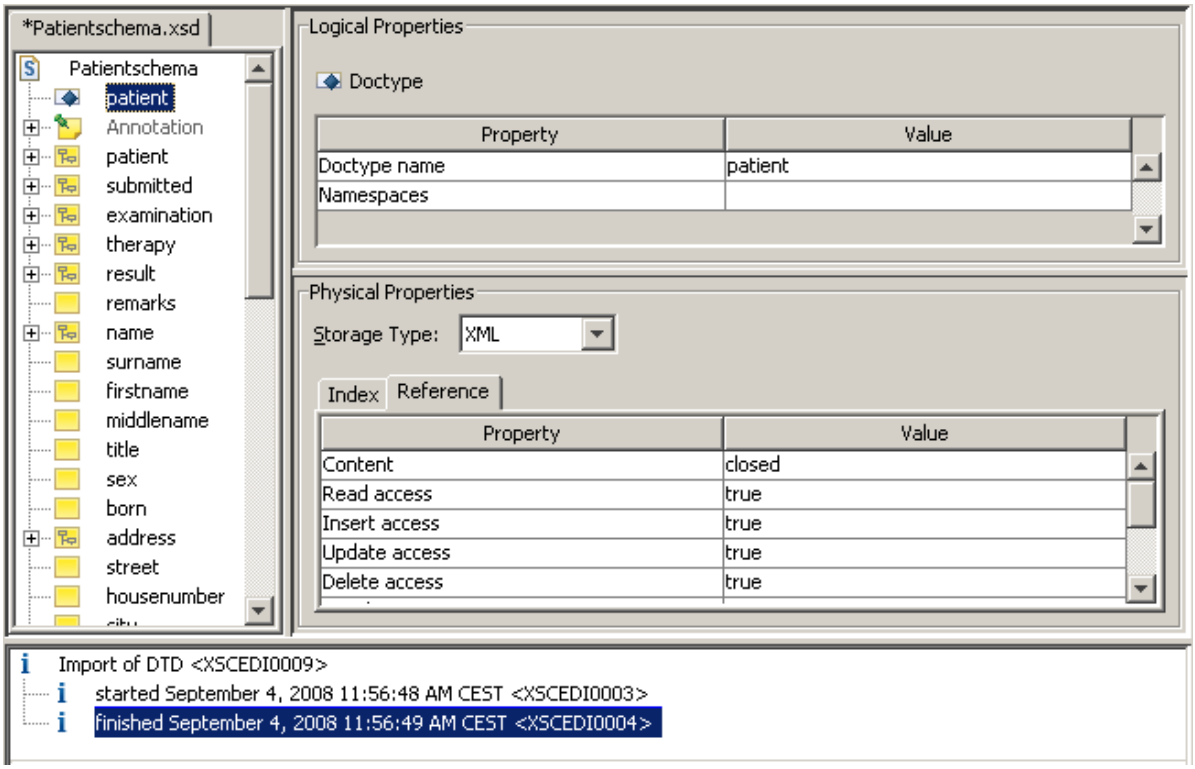
First, select the node **Patientschema** in the the view.

7 From the **File** menu, choose **Import DTD...**

The dialog box **Import DTD** opens.

8 Browse to the DTD file you want to load and define a schema for. The examples are located in several subdirectories of *<TaminoDocRootDir>/examples*, where *<TaminoDocRootDir>* is the starting directory of the product documentation. In this example, browse to *<TaminoDocRoot-Dir>/examples/patient/patient.dtd*. Choose the **Import** button to import the DTD.

The DTD is automatically converted to XML Schema format and loaded into the Schema Editor. On the left, the editor displays a tree view of the newly generated schema. The right part shows the logical and physical properties. In the bottom frame, certain status messages are displayed.



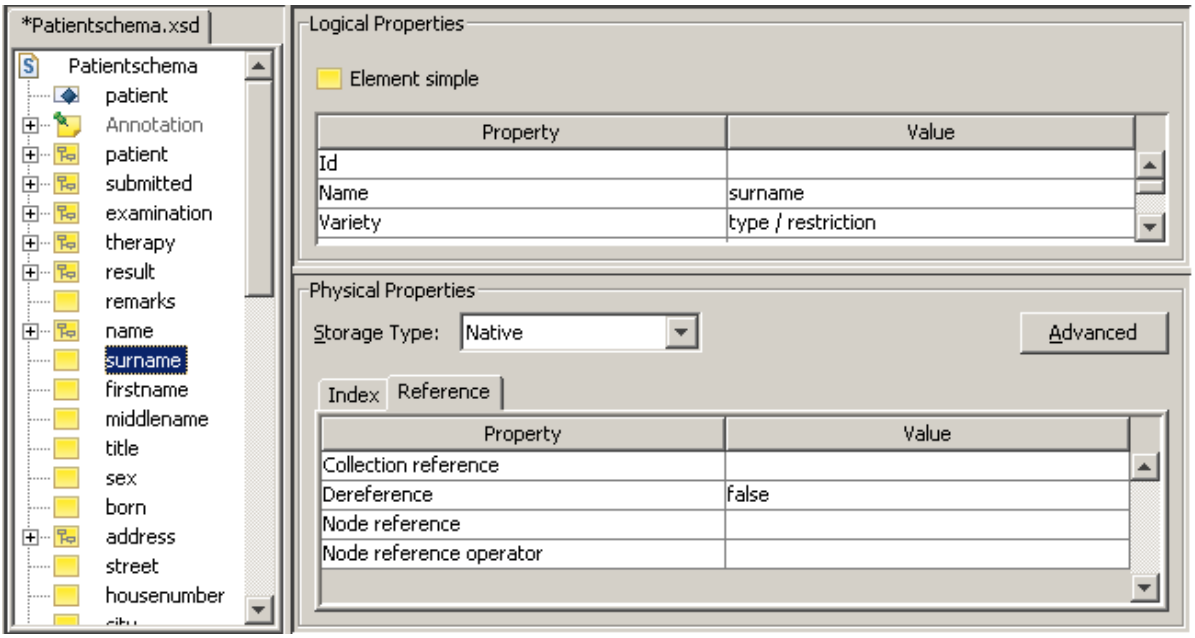
Tip: If you wish, you may save the schema file locally (use **Save as...** in the **File** menu) and inspect it with your favorite text editor. It should contain a doctype declaration and a "skeleton" schema with logical properties (XSD constructs), but without physical properties (TSD constructs). A sample schema file can also be found in *<TaminoDocRoot-Dir>/examples/patient/HospitalSchema.tsd*.

The final step is to define physical properties for some of the elements. For this example, we will define properties for the elements `surname` and `born`. Later in this chapter, they will be used for queries.

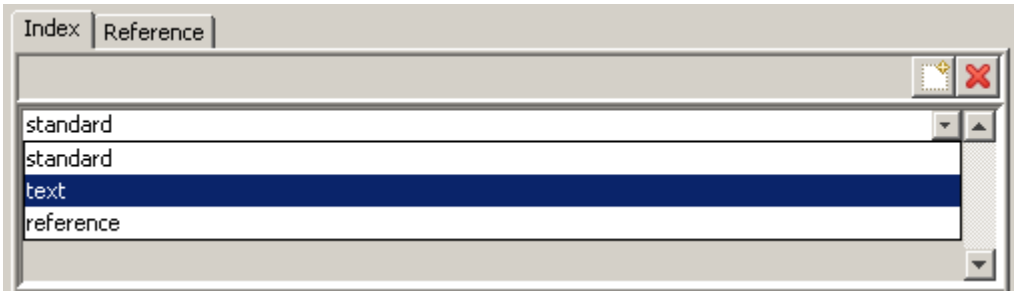
To define physical properties:

1 Choose the node for which you want to define properties. In this example, choose `surname` in the tree view on the left-hand side of the Schema Editor window.

On the right-hand side, the logical and physical properties for this node are displayed:



2 Under **Physical Properties**, choose the **Index** tab. Choose the **Add New Index** icon ( ), to create a new index. By default, an index of type "standard" is displayed. Open the drop-down list for this field and choose `text`:



This will create a text index on the element `surname`.

3 Next, choose the element `born`. Again, the properties of this element are displayed in the right hand window of the Schema Editor. For this element, choose the value `standard` for the index. Also, change the logical property `Data type` to the value `xs:integer` by scrolling through the list of values in the **Logical Properties** window.

You now have loaded a DTD into the Tamino Schema Editor, generated a schema based on the DTD, and defined some properties for some of the elements. Later, this is important for a quick retrieval of your data. In the next step, you will learn how to define the schema as such to Tamino. But first, save the schema as an XML file:

To save a schema as XML:

1 Choose **File > Save As...**.

The dialog box **Save As** appears.

2 In the dialog box, browse to the directory you want the schema to be saved in, for example *C:\temp* on Windows or */tmp* on UNIX, and enter a name for the schema. Use the file extension *.tsd*, for example *Patientschema.tsd*. Choose **Save**.

More details about the Schema Editor are available in the documentation for Tamino Schema Definition and the Tamino Schema Editor.

7 Defining a Schema

You have learned how to generate a schema (based on a DTD), how to modify and save it. In this section, you will learn how to define a schema to your current Tamino database so that it becomes a part of Tamino's Data Map. The schema is then available to other Tamino components. Defining a schema is a prerequisite for loading and retrieving XML objects.

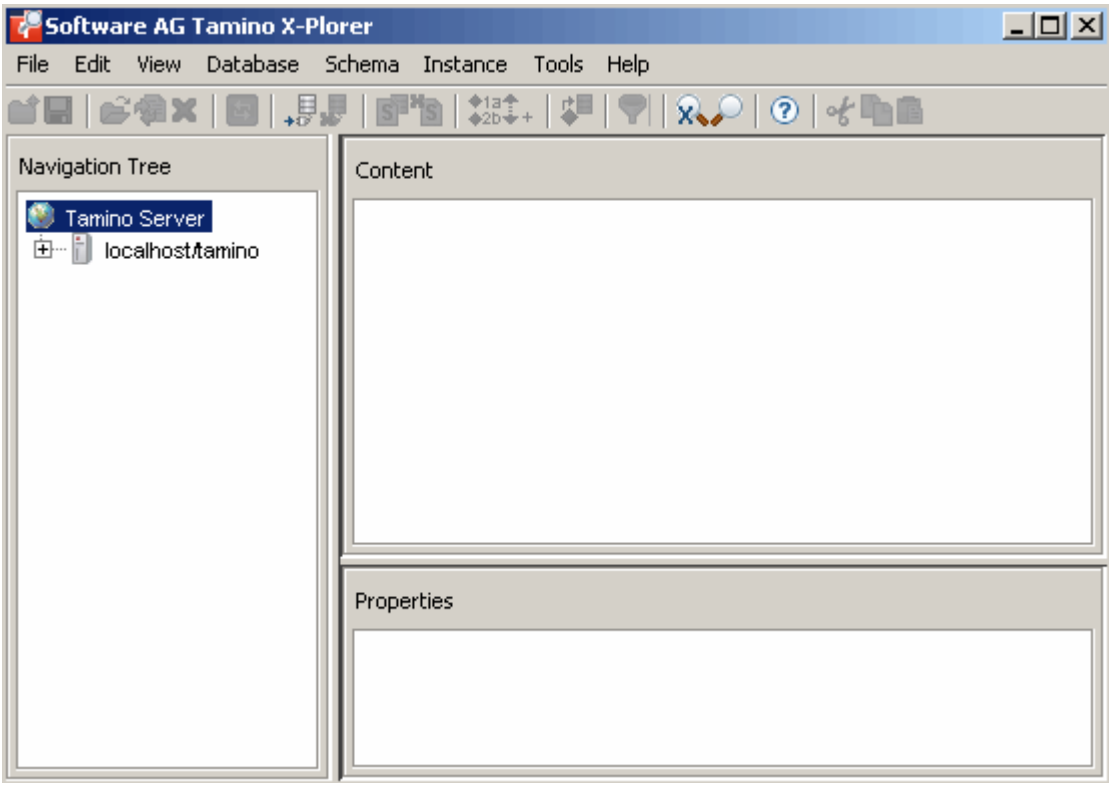
Although there are several ways of defining schemas and loading objects, the most comfortable one is to use the Tamino X-Plorer. The Tamino X-Plorer allows you to perform various database operations such as defining collections and schemas, loading and retrieving data as well as many other actions with the help of an easy-to-use graphical interface.

To start the Tamino X-Plorer

- Windows: choose the shortcut **Tamino X-Plorer** in the Tamino program group that is available from the **Start** button.

  UNIX: Start the following script from the command line: `inoxplorer.sh`.

  The Tamino X-Plorer is displayed:



The X-Plorer consists of three panes, the navigation tree on the left, the (currently empty) Content pane and the Properties pane on the right. The navigation tree shows databases of Tamino servers to which you have previously connected. The Content pane shows the content of the XML object that is currently selected in the navigation tree. The Properties pane shows properties (metadata) for the currently selected object. For the following examples, we will use the predefined Tamino sample schema *HospitalSchema.tsd*, which can be found in the directory *<TaminoDocRootDir>/examples/patient*. The first step will be to define this schema to our database "my-database". If you have followed the preceding steps in this *Getting Started*, you can also use your own schema *Patientschema.tsd*, which has already been created in the previous section. Both files (*Patientschema.tsd* and *HospitalSchema.tsd*) should be identical.

Note: Do not define both schemas, *HospitalSchema.tsd* AND *Patientschema.tsd* to the database, as they define the same doctype in the same collection. This will lead to an error, since doctypes need to be unique.

To define the schema *HospitalSchema.tsd* to the database "my-database"

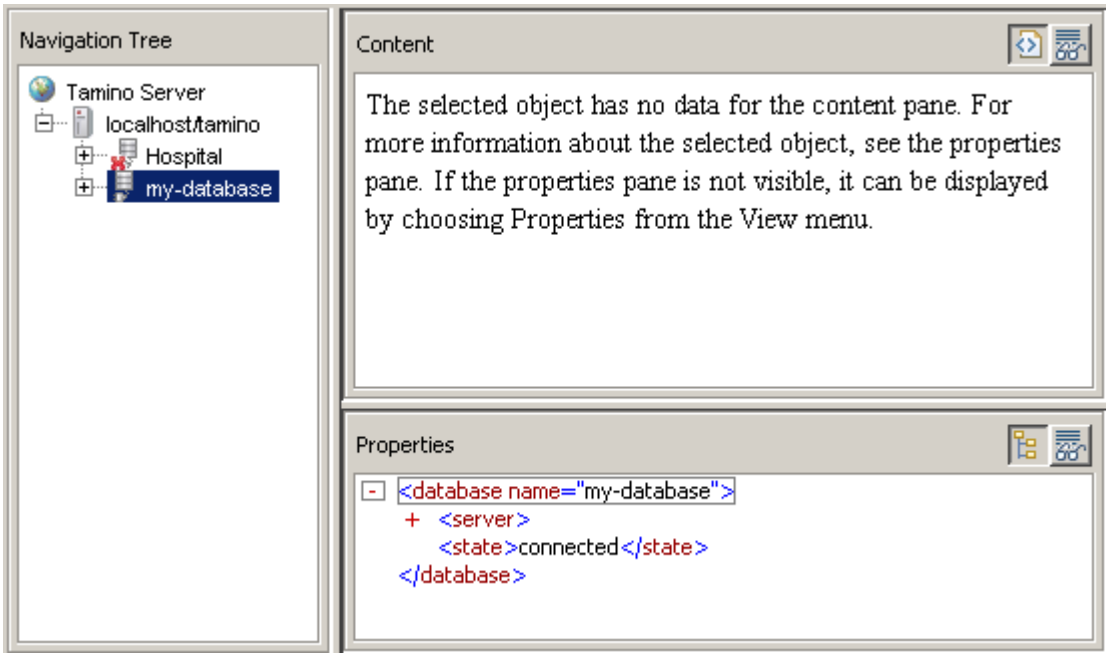
1 Expand the navigation tree to see the list of known servers. To do so, click on the plus signs in the navigation tree until the names of the known databases are displayed. If you do not see the database *my-database* that we created previously, choose **Database > Connect** from the main menu to start the **Connect to** dialog. In this dialog, choose the **Select** button next to

the **Database** field in order to display the names of the available databases. Choose *my-database* from the list, then choose the **Connect** button.

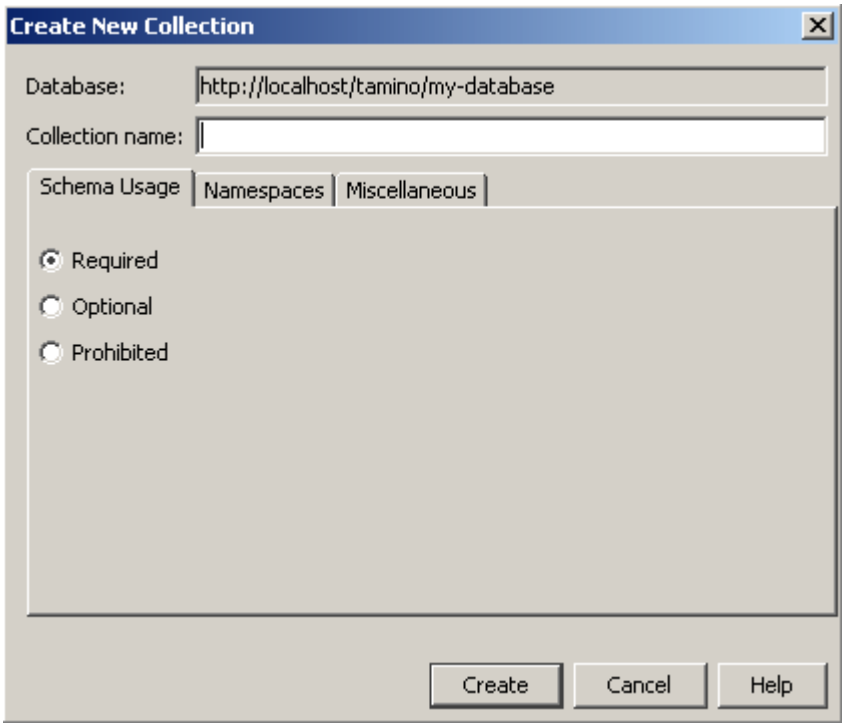
For detailed information, see the section *Connecting to a Tamino Database* in the Tamino X-Plorer documentation.

Now the database *my-database* should be visible in the navigation tree.

2 In the navigation tree, select the database "my-database". Information such as the following appears in the X-Plorer panes:

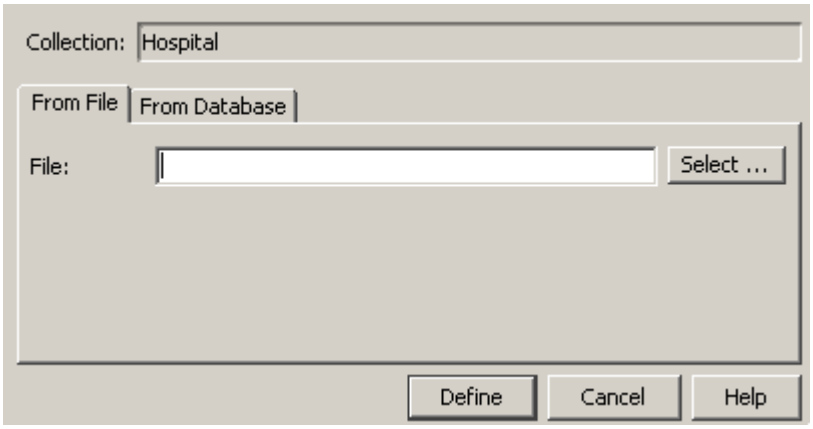


3 Next, you need to define a collection to which the schema will be defined. To do so, select the entry "my-database" in the navigation tree, then choose **File > New collection**. The dialog box **Create New Collection** is displayed:



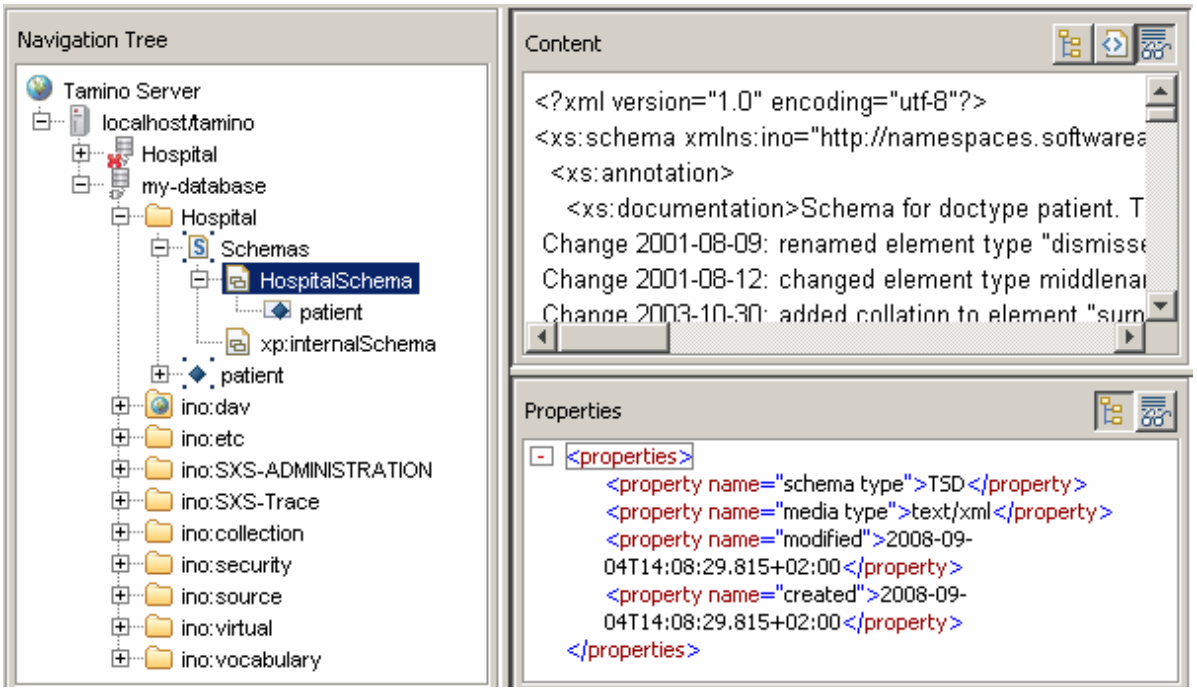
4 Enter a collection name, in this case `Hospital`, leave everything else in the dialog box as is, and choose the **Create** button. The new collection is displayed in the navigation tree under "my-database".

5 Select the collection `Hospital` in the navigation tree and choose **Schema > Define Schema**. The **Define Schema** dialog box is displayed:



6 In the **From File** tab, use the **Select** button to browse to the predefined schema file *HospitalSchema.tsd*, located in the directory *<TaminoDocRootDir>/examples/patient*. Choose the **Define** button to define the schema to the database "my-database".

In the navigation tree, "HospitalSchema" is now displayed under "Schemas" when node "Hospital" is expanded. If you select it, the schema is displayed in the content pane, and information about it appears in the Properties pane.



Now you have successfully defined a schema to the collection `Hospital` in the Tamino database "my-database". The next step is to load data into the database. In the following, you will learn how to load XML objects and non-XML objects into the database.

8 Loading XML Objects into the Database

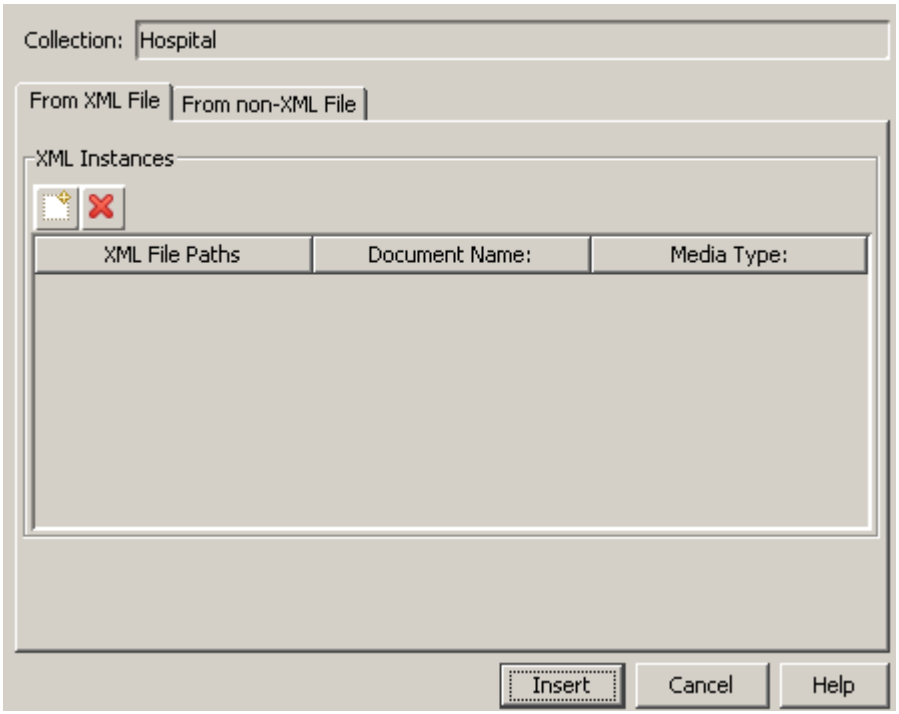
In this chapter, we will load some data into the database "my-database". We have prepared the sample data files *atkins.xml* and *bloggs.xml* for you, so that you do not have to bother about creating XML files yourself. Each of these files contains one instance of the doctype *patient*, which is defined in the previously mentioned schema *HospitalSchema*. The sample data files are located in the directory `<TaminoDocRootDir>`*/examples/patient*.

To load XML objects into the database

1 In the Tamino X-Plorer, navigate to the collection `Hospital` and select it.

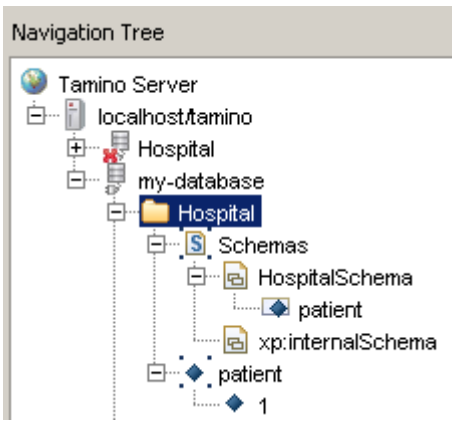
2 From the **Instance** menu, choose **Insert Instance**.

The Insert Instance dialog box is displayed:

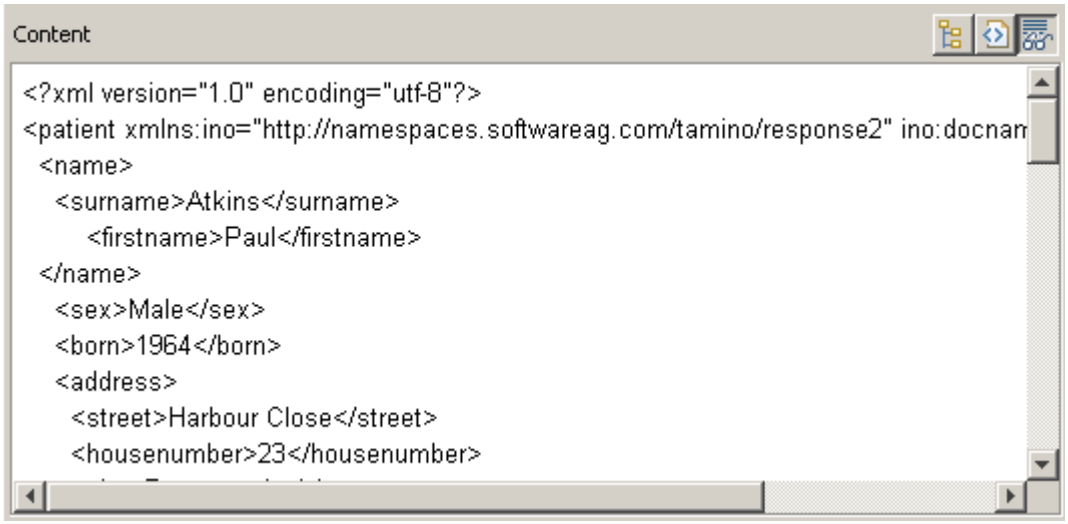


3 On the **From XML File** tab, choose the browse button ( ) and browse to the sample data file *atkins.xml*, which is located in the documentation examples directory *<TaminoDocRoot-Dir>/examples/patient*. Choose the **Insert** button to load the file into the database "my-database".

In the navigation tree, the loaded data instance is displayed in the following way:

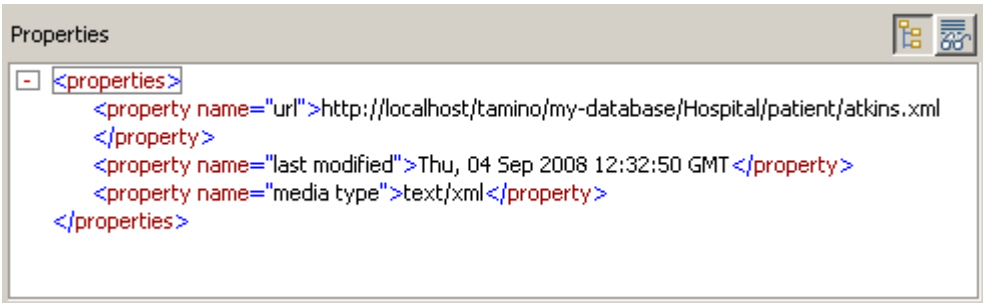


The data for patient *Atkins* is represented in the navigation tree by the number "1" underneath the *patient* node. If you select it, you can see the corresponding XML data in the plain text view in the Content pane:



(The icons in the Content pane allow you to choose between plain text view, HTML view and tree view.)

In the properties pane, metadata is displayed:



For XQuery examples that will follow in the section ***Retrieving Objects from the Database Using XQuery***, now go back to step 1 and also load the XML data set for Mr. Bloggs with the filename *bloggs.xml*.

The method described here is one of many ways of loading data into Tamino. Another way of loading objects that are structured according to a schema is to use a client programming interface. Tamino offers client APIs for Java and C. Tamino also offers a programming interface on the basis of the X-Machine programming language. To access the documentation for these APIs, refer to the documentation overview page. An additional method is to use the Tamino Data Loader. Refer to the Tamino Data Loader documentation for details.

9 Loading non-XML Objects into the Database

You can also load documents that are not XML documents, such as plain text or graphic files. For such documents, Tamino sets the default collection name to `ino:etc` and the default doctype name to `ino:nonXML`, in other words, all non-XML documents that are loaded without a schema will be stored in this collection and doctype, if nothing else is specified. However, non-XML documents can reside in any collection and doctype you specify. It is also possible to load non-XML data using a schema. For more information, see the section Storing Non-XML Objects in Tamino in the Tamino Schema Definition documentation.

As an example, we will now load the sample non-XML file *fevercurve.gif* into the database.

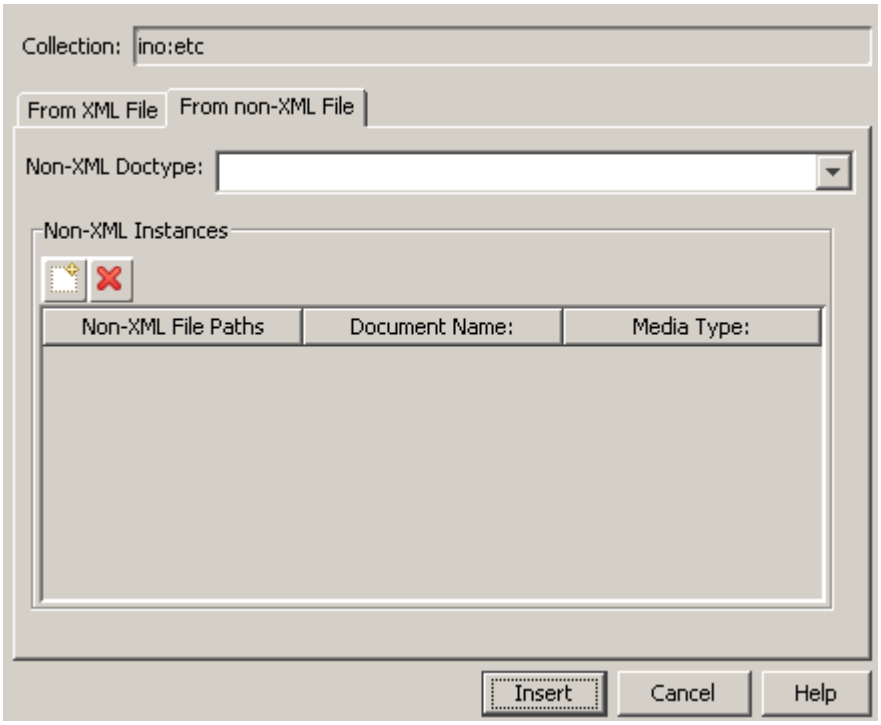
To load a non-XML file

1 In the navigation tree of the X-Plorer, select the collection `ino:etc`.

2 From the **Instance** menu, choose **Insert Instance**.

The Insert Instance dialog box is displayed.

3 Select the **From non-XML Files** tab:

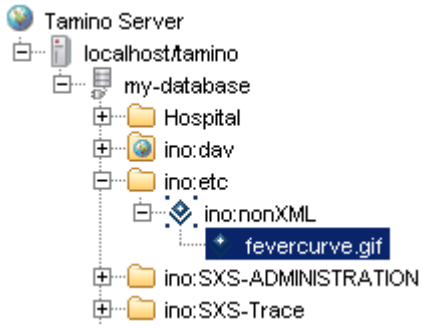


4 In the Non-XML Doctype field, enter *ino:nonXML*, or choose it from the selection list.

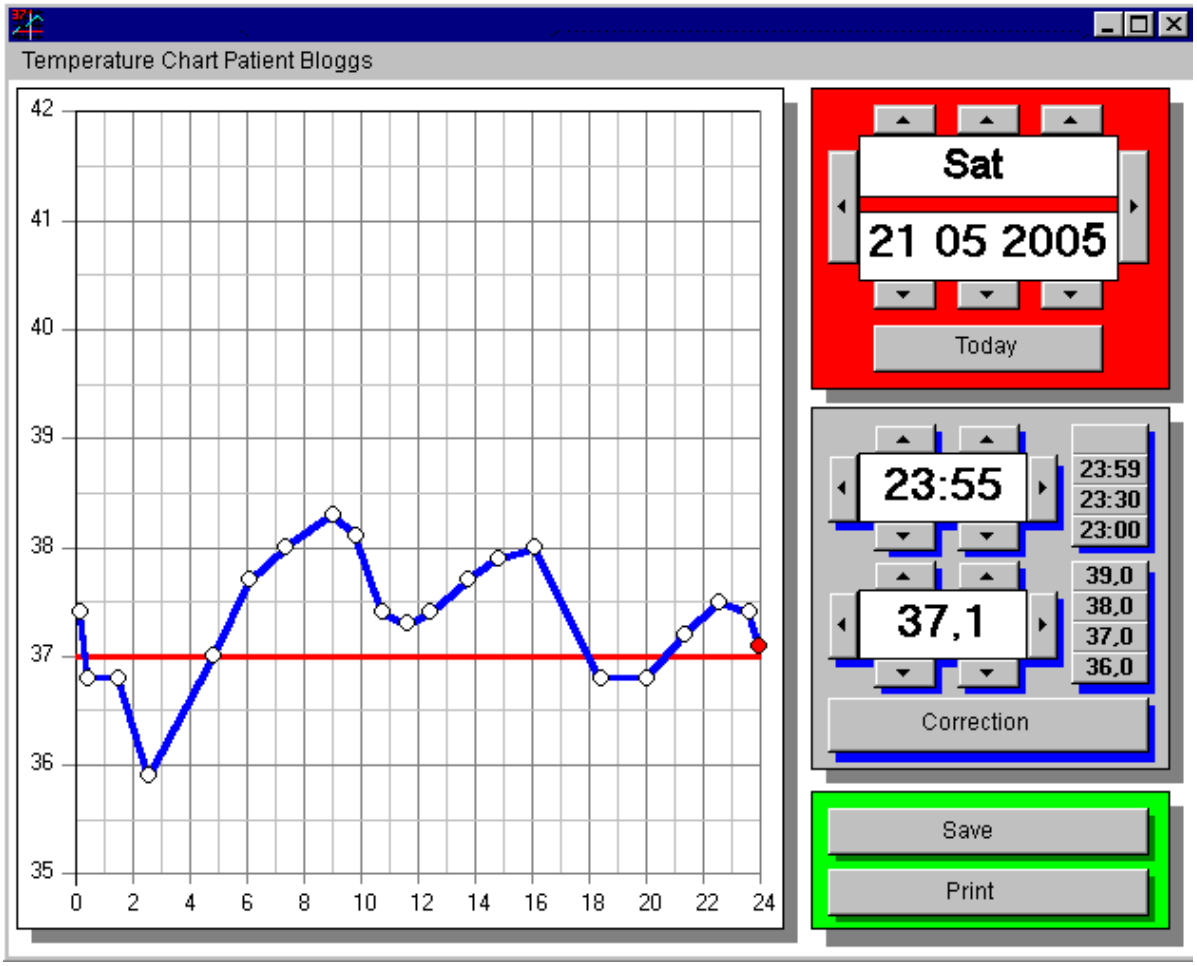
5 Use the button to browse to the non-XML file to be inserted. For this exercise, use the sample graphics file *fevercurve.gif* that is located in the same directory as the sample XML data for the patients Bloggs and Atkins: *<TaminoDocRootDir>/examples/patient*.

6 Choose the **Insert** button.

The file is now inserted into the navigation tree:



Select it to view it in the Content pane (ensure that the **Image View** button is selected in the Content pane):


Temperature Chart Patient Bloggs
42
41
40
39
38
37
36
35
0
2
4
6
8
10
12
14
16
18
20
22
24
Sat
21 05 2005
Today
23:55
23:59
23:30
23:00
37,1
39,0
38,0
37,0
36,0
Correction
Save
Print

10 Retrieving Objects from the Database Using XQuery

Once you have loaded objects into the database, you can retrieve the objects using XQuery statements. XQuery is the W3C standard query language for structured documents. Tamino allows you to use XQuery for performing queries on XML (and also non-XML) objects. XQuery allows you not only to retrieve database contents but also to compose your query result using constructors.

In addition to XQuery, Tamino still supports the query language X-Query (written with a hyphen) used in former versions of Tamino. See the X-Query User Guide for detailed information.

In the following, some simple examples of query expressions with XQuery are provided to familiarize you with the basic retrieval process. For the examples, use the data you loaded into the database in the preceding sections (Atkins and Bloggs).

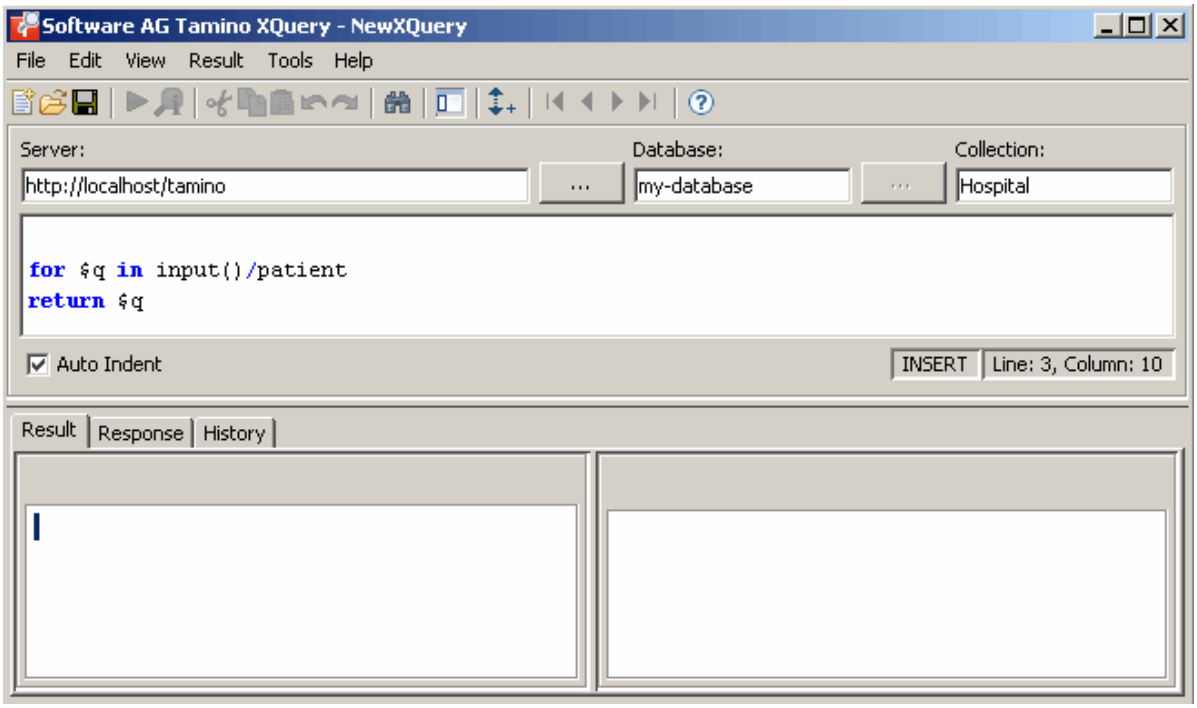
The easiest way to query data is to use the XQuery Tool. You can open it from the Tamino X-Plorer.

To open the XQuery Tool from the Tamino X-Plorer

1 In the navigation tree, select the doctype `patient`.

2 From the **Tools** menu, choose **Query > XQuery**.

The Tamino XQuery Tool is displayed, and the XQuery code required to return all instances of the selected doctype *patient* is provided automatically:



The XQuery window contains all information that is required to execute a query: server, database, collection and the query code itself.

Before you try the examples, please delete the predefined query code, since we will use our own examples.

3 Under **Tools > Options > Module** in the XQuery tool, ensure that the box **Module management mode** is not checked.

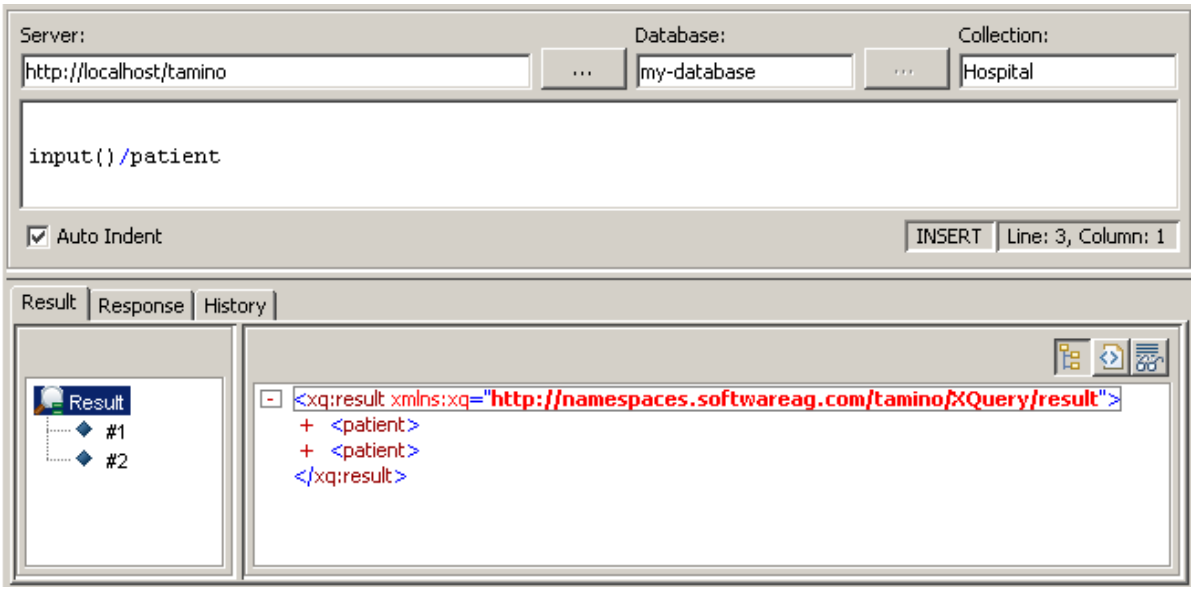
To find all elements of the type "patient" in the current collection:

- Enter the following in the XQuery field:

```
input()/patient
```

Choose the **Execute XML Query** button in the toolbar.

The query response is returned by Tamino as an XML object in the result window of the Tamino XQuery Tool. You should get a query result, listing all patients (in this case patients Atkins and Bloggs; to display more information about the patients, expand the plus-sign):



The path operator (/) in the query indicates that the selection starts at the document node (the first node in the tree). The expression "input()" denotes the current collection and returns documents stored in the collection `Hospital` whose document node is `patient`.

To find all patients born after a certain date, using data type xs:integer:

- Enter the following in the XQuery field:

```
for $a in input()/patient
where $a/ born > 1960
return $a
```

  Choose the **Execute XML Query** button.

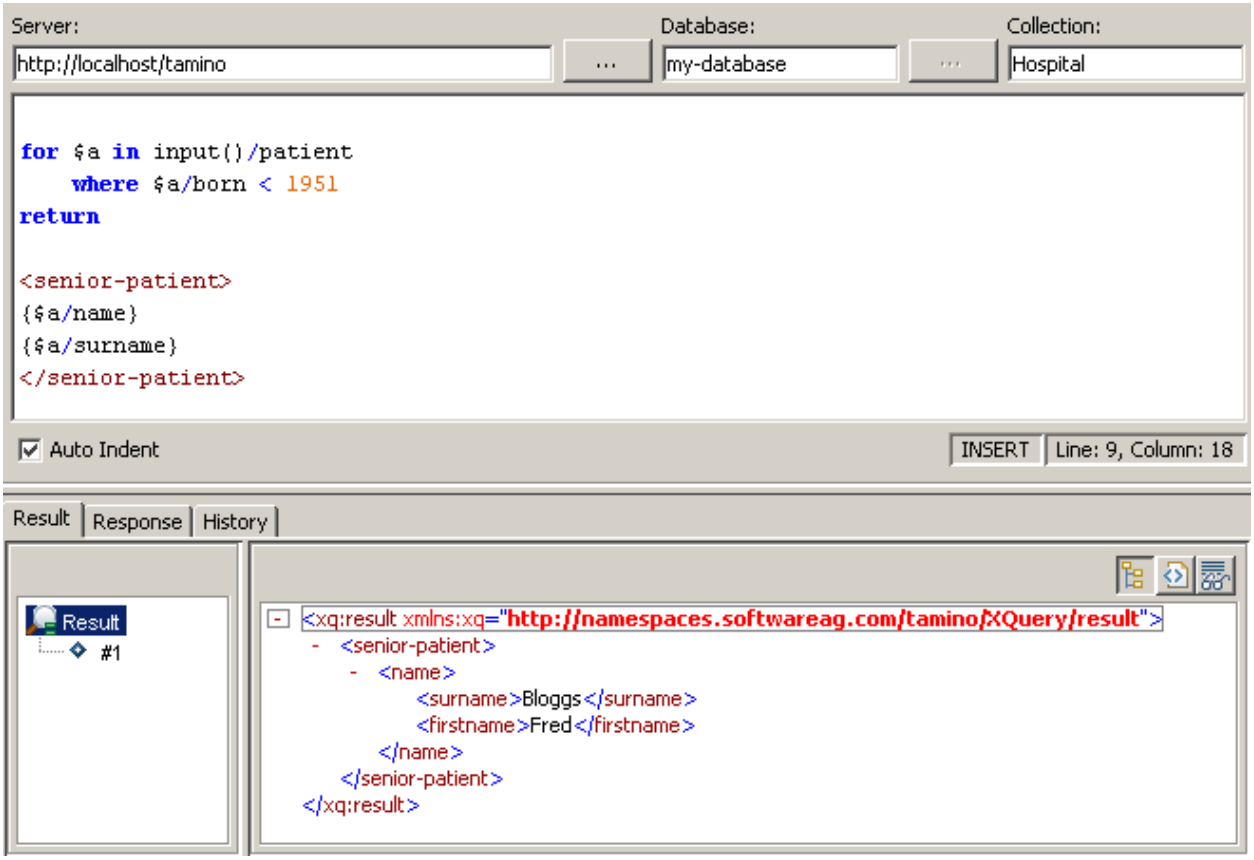
  In the output window, you should get the data from patients born after 1960 (in this case patient Atkins).

  Remember that in the first part of this *Getting Started*, we put a standard search index on the `born` element and defined the data type `xs:integer` for it, so that a quick search and comparison can be performed on integer values. (However, a search would also be performed without the index).

You can construct new elements (in the following example, the element `senior-patient` is constructed) and combine your results with the following query:

```
for $a in input()/patient
where $a/born < 1951
return
<senior-patient>
{ $a/name }
{ $a/surname }
</senior-patient>
```

You should get the following result:



In the return clause, an element constructor is used simply by writing the appropriate tags. As child elements, two expressions are enclosed in braces, which means they need to be evaluated. This query returns a list of senior patients with name and surname (in this case patient Bloggs).

See the XQuery documentation for more information and examples about the XQuery language. For detailed information about the XQuery tool, see the documentation about the Tamino X-Plorer, section *Using the Tamino X-Query Tool*.

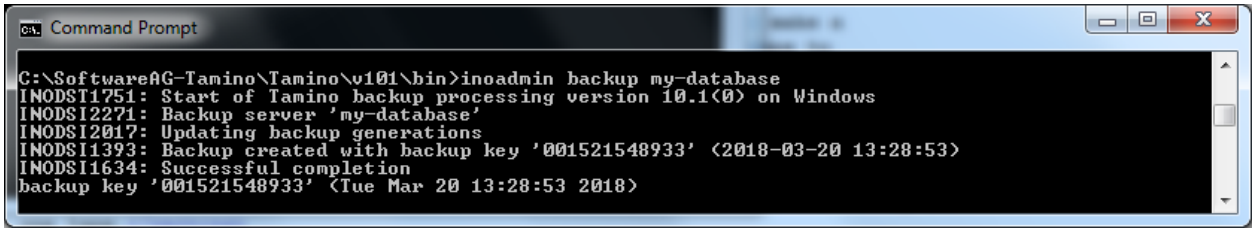
11 Backup and Restore

An important part of database administration is to make regular backups of your database. Database Backup and Restore functions are available via the **inoadmin** commandline tool. The **backup** command makes a copy of the database. The **restore** command restores the database to the state it was in when you made the backup. Restore is generally executed together with a Recover process, in which all changes that were made to the database after the backup are re-applied. During the Recover process, log files (log spaces) that record all completed database transactions that have occurred since the most recent backup are used as input files. In this way, all data up to the most recently completed transaction can be restored.

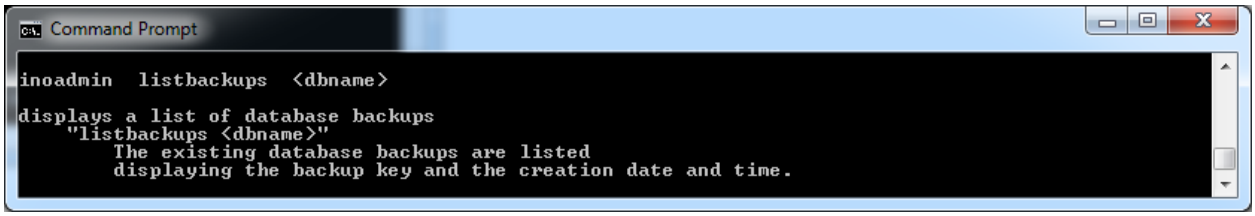
Backup

To create a backup use the inoadmin **backup** command.

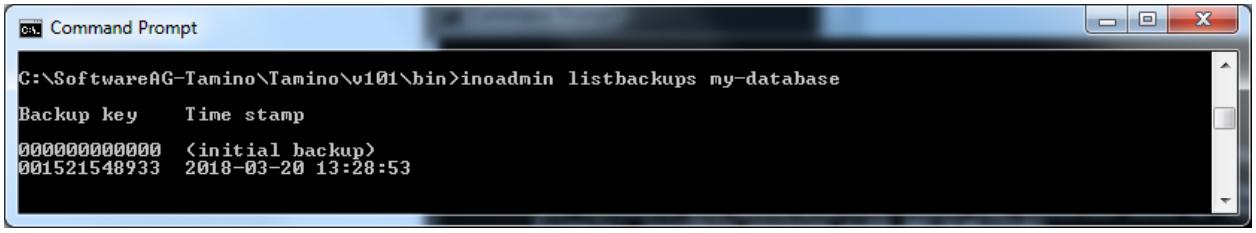
If no location is provided the backup is stored at the initial location. The database to be backed up needs to be started.



The inoadmin **listbackups** command shows all currently existing backups of a database.



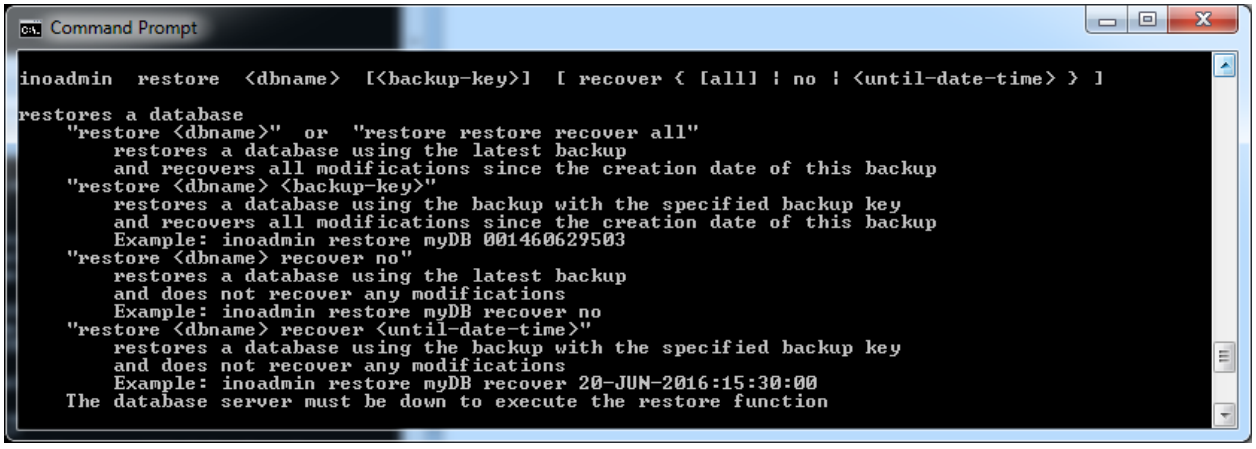
So, having backed up the database the new backup appears as a result of the inoadmin **listbackups** command.



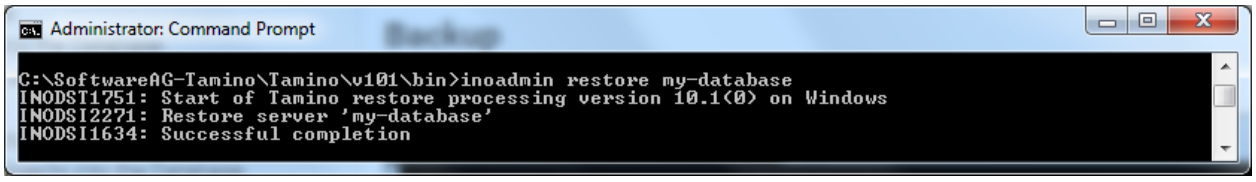
Besides the new backup there is allways the initial backup that is created with the database itself.

Restore and Recover

The inoadmin **restore** command allows to restore the database to a previous state or to the actual state in case of failure.



To restore and recover a database from a backup the database must be down. Just issuing the command with <dbname> as the only parameter re-establishes the database by using the latest backup and applies all modifications stored in the log files.



12 Where to go from here

Congratulations! You have mastered your first steps with Tamino, and hence have gathered hands-on experience with a complete data management system for exchanging data and integrating applications on an XML basis. You now know how to create, start and stop a database, to load and retrieve XML as well as non-XML objects into the database, and to save your database with the backup and restore functions. Next, we recommend you to get more detailed information about the Tamino Manager and the Tamino Schema Editor, which you will find in the respective documentation.

We also recommend you to become a member of the **Tamino Developer Community**, a web site which focuses on providing valuable information, guidance and help to developers who use the Tamino Platform to build software applications, solutions or products.

Index