Tamino

Tamino Interactive Interface

Version 9.7

April 2015

WEBMETHODS

This document applies to Tamino Version 9.7.

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

Document ID: INS-TII-97-20160318

Table of Contents

Tamino Interactive Interface

The Tamino Interactive Interface is an effective browser-based interface to the Tamino Server. It allows you to define schemas, load, query and delete data, undefine schemas, as well as diagnose the database.

This document covers the following topics:

1 Introducing the Tamino Interactive Interface

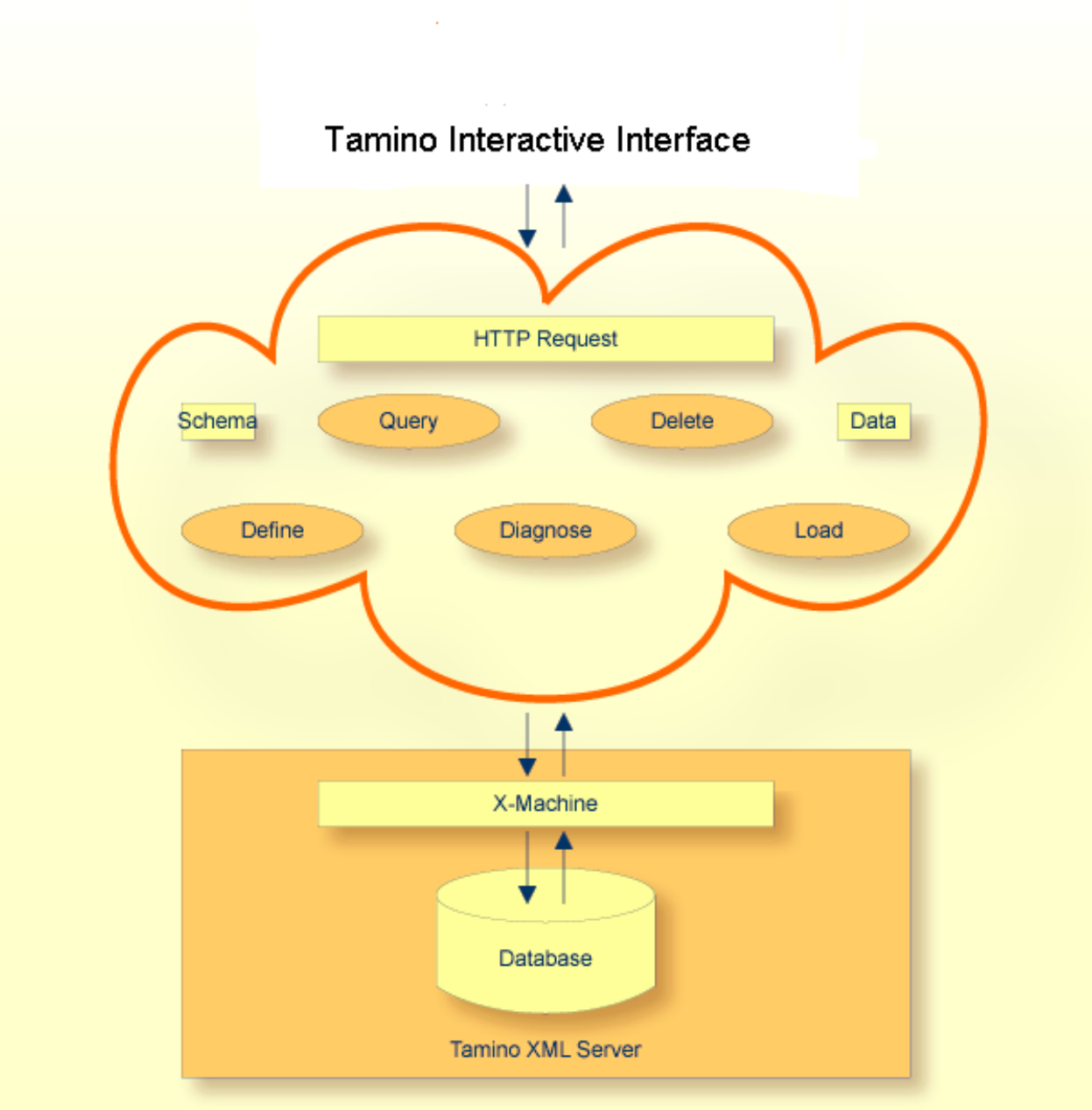
The Tamino Interactive Interface allows you to access Tamino using a web interface. Thus it is easy to send commands such as queries, deletions, updates etc. via HTTP-requests to Tamino's X-Machine, without having to bother about Tamino internals. You can query a database using Tamino X-Query or the new Tamino XQuery language (as proposed by the W3C), delete documents using Tamino X-Query, load the content of a file to insert or update a document, define or update a Tamino schema, undefine collections, schemas or doctypes, diagnose a database, and enter authentication information with the help of intuitive browser-based input fields.

Note: The Tamino Interactive Interface is not intended for loading large amounts of data into Tamino. For this purpose, use the Tamino Data Loader. For more information, see the documentation about the *Tamino Data Loaders*.

Requests from the Tamino Interactive Interface to the Tamino Server cause the Tamino Server to return an XML response document. To display this response document properly, the Tamino Interactive Interface must be running in an XML-capable browser. If you are not using an XML-capable browser, the actions listed above can still be performed without restriction, but the response document will not be displayed properly in the browser.

The Tamino Interactive Interface can use the standard keyboard access of your browser. For details on keyboard access not mentioned in this documentation, consult the documentation of your individual operating system or software products.

The features of the Tamino Interactive Interface are summarized in the following graphic:


Tamino Interactive Interface
HTTP Request
Schema
Query
Delete
Data
Define
Diagnose
Load
X-Machine
Database
Tamino XML Server

2 Starting the Tamino Interactive Interface

For information about starting the Tamino Interactive Interface, see the startup procedures for your platform as described in the chapter *Starting Tamino*.

Under Windows, you can start the Tamino Interactive Interface from the **Start** button by selecting the appropriate shortcut from the **Tamino Tools** menu under the **Tamino** program group.

Under UNIX, you can use the shell script *inoint.sh* to start the Tamino Interactive Interface. The location of this script is added to the `PATH` environment variable during the Tamino installation.If Tamino is installed on a remote machine and you want to access the Tamino Server installation from your local machine, perform one of the following steps before using the shell script:

- Copy the files from the directory containing the Interactive Interface to the document root served by the Web server (this is the value of the parameter `DocumentRoot` in the web server's configuration file *httpd.conf*), using a command of the following form:

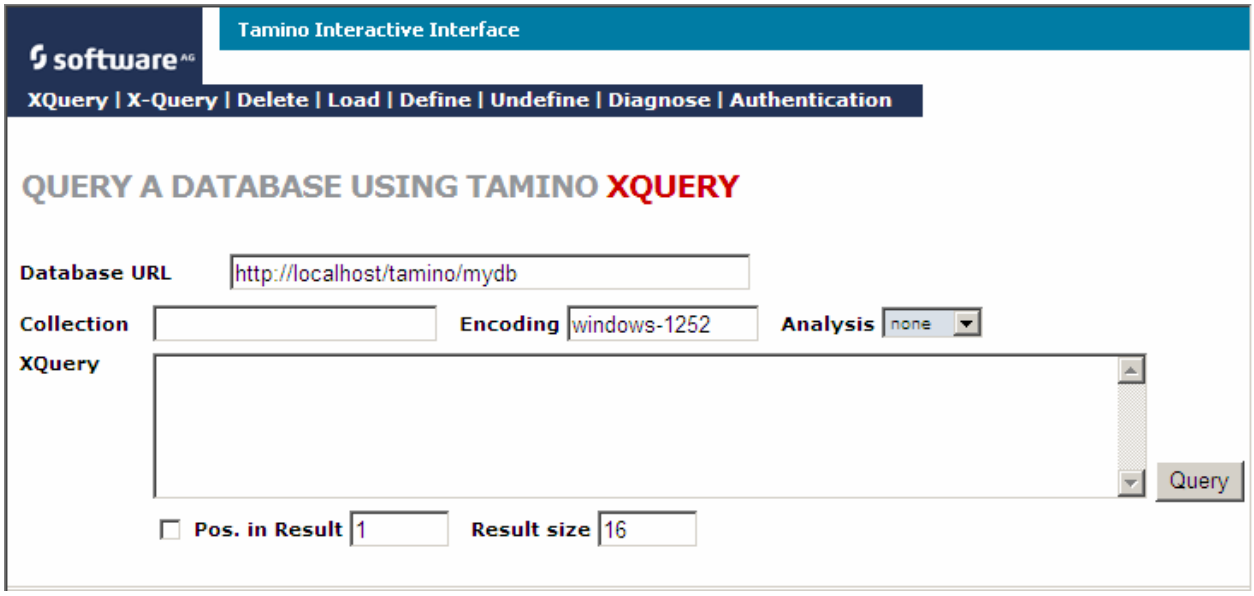
```
$ cp -r  <TaminoInstallDir>/X-Tools/Tamino_Interactive_Interface  <DocumentRoot> ↩
```

  Note: `<TaminoInstallDir>` is the directory where the Tamino Version is installed, on Windows e.g. *C:\Programs\Software AG\Tamino\Tamino 4.2*.

- Edit the Apache configuration file and add an alias of the form:

```
Alias /tii/ "<TaminoInstallDir>/X_Tools/Tamino_Interactive_Interface"
```

Once you have started the Tamino Interactive Interface, a screen similar to the following is displayed:



Note: Your display may differ from the one shown above depending on what browser you use. The button **Favorite**, for example, is only displayed when the Tamino Interactive Interface was loaded with Microsoft's Internet Explorer via HTTP.

3 Common Elements of the Tamino Interactive Interface

The Tamino Interactive Interface consists of several tabs. Each tab displays a form with input fields necessary for executing a command to Tamino. When you enter data into the input fields and send a command to Tamino, the result is displayed in the result window in the lower part of the Tamino Interactive Interface. The content of the fields **Database URL**, **Collection**, and **Encoding** are shared by all forms that need one or more of these input fields. Help is available on each form, if the Tamino Interactive Interface is loaded via HTTP.

This chapter describes the following common input fields:

Database URL

Here you specify the URL of the (existing) Tamino database that you wish to access. Enter a valid URL for a Tamino database, for example `http://localhost/tamino/mydb`. Database names are case sensitive. If you want to switch to a different Tamino database, enter a new URL here. Information on how to create Tamino databases can be found in the Tamino Manager documentation.

Note: For the examples described in the section *Working with the Tamino Interactive Interface*, the database you specify needs to be started. For information on how to start a database, see the Tamino Manager documentation.

Collection

Here you specify the (existing) collection that you wish to access. Enter a valid name for the collection. Collections are defined within the Tamino Schema. Information about creating and naming collections can be found in the documentation about the *Tamino Schema Editor*.

Encoding

Here you specify the character encoding used when sending a Query, Delete or Undefine request to Tamino. In this case, the request is sent to Tamino with the encoding you have specified in the **Encoding** field.

If you send a Load or Define request, the XML document itself possesses encoding information within its header. Subsequently, the entry in the **Encoding** field is ignored. If no encoding is specified, "ISO-8859-1" is set as default.

With Microsoft Internet Explorer, this field will be initialized with the default encoding you have specified within your browser settings. In general, the encodings accepted by the Tamino Interactive Interface depend on the encodings supported by the browser.

Favorite

With this button, you can save the current settings as a bookmark. Currently, this option is only available if your browser is Microsoft's Internet Explorer and you loaded the Tamino Interactive Interface via HTTP. If you choose this button, the current settings of the Tamino Interactive Interface are saved as a bookmark. You will be prompted to enter a name and location for the shortcut.

The parameters will automatically be read into the URL. The following parameters are available:

Parametername	Description
`DatabaseURL`	The URL of the Tamino database, e.g.: `http://localhost/tamino/mydb`
`Collection`	The name of the collection.
`Query`	A query string.
`Explain`	Switch the usage of the ino:explain function on or off. See ***Analysis: Request information concerning the query***.
`Resultsize`	The result size as used in the Tamino Interactive Interface
`Delete`	The delete string.
`Undefine`	The undefine string (`collection-name` or `collection-name/doctype-name`).
`Encoding`	The encoding value, e.g. `utf-8` or `iso-8859-1`
`Runcmd`	Executes the specified command on startup of the Tamino Interactive Interface. Possible values are: ■ `query` = execute the query string specified with the `query` parameter ■ `xquery` = execute XQuery ■ `delete` = execute the delete string specified with the `delete` parameter ■ `undefine` = execute the undefine string specified with the `undefine` parameter. You still will be prompted to confirm this action. ■ `diagnose` **Note:** This parameter does not check other parameters, e.g. it is assumed that a query string exists, if the query command is triggered.

Note that the parameter `Runcmd` acts like the **OK** button in the Tamino Interactive Interface. Press the button **Favorite**, then edit the URL and append the parameter `&RUNCMD=...` manually to the end of the URL. If you do so, the command is executed when you start the Tamino Interactive Interface again.

Also note that the parameter names are not case sensitive, but the values are.

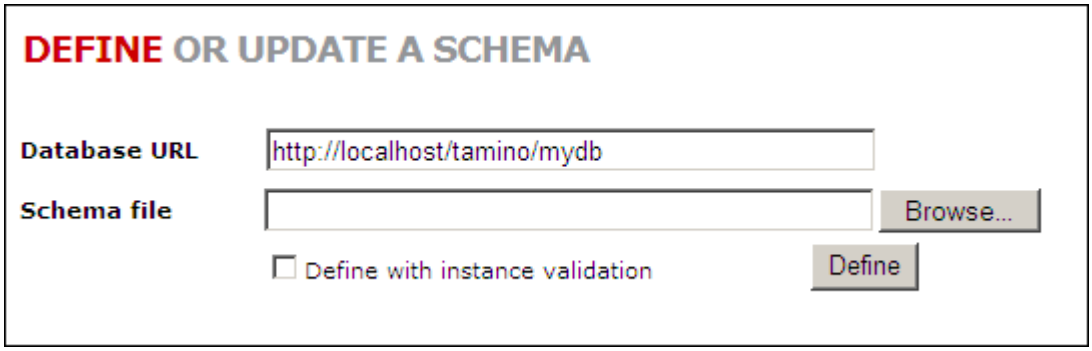
4 Working with the Tamino Interactive Interface

In this chapter, you will find information about the tasks you can accomplish with the Tamino Interactive Interface, illustrated by an example.

The following topics are covered:

Define or Update a Schema

The first thing you may want to do when working with a Tamino database is to define a schema. Start the Tamino Interactive Interface and choose the form **Define or update a schema** by selecting the **Define** tab. A screen similar to the following is displayed:



Enter the Database URL (see *Database URL*). Go to the field **Schema file**. In this field, you specify a file that contains a schema definition. A schema definition describes the structure, but not the contents, of a set of objects that will be loaded into the database. You can either type in the filename and full path name directly, or you can use the **Browse** button that is located at the end of the field to locate the file that you want.

By default, the schema is defined without validation. If you check the box **Define with instance validation**, the schema is defined *and* validated.

Note: A sample schema file named *HospitalSchema.tsd* is provided in *<TaminoDocRootDir>/examples/patient*, where *<TaminoDocRootDir>* is the starting directory of the product documentation. If you want to follow the example given in this section, use it as your sample schema.

When you have specified the schema definition file, choose the **Define** button. This loads the schema into the selected database and collection.

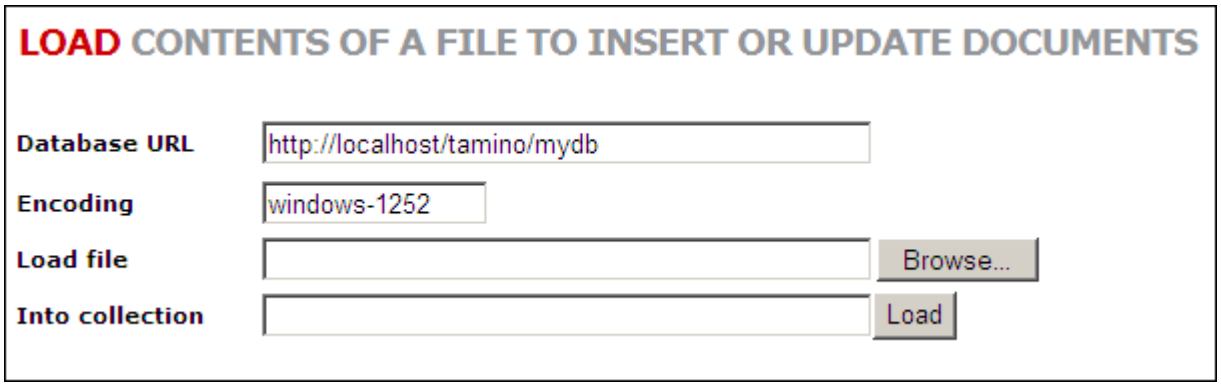
For a full description of creating schema definitions, refer to the documentation about the *Tamino Schema Editor*.

Note: Netscape versions 6 and above determine the content type of form data by the file extension. Schemas named *.tsd* will be submitted with the content-type application/octet-stream which Tamino will not recognize as valid schema data. Use the following work-around: For Netscape 6.x, rename the schema files to *.xml*. For Netscape 7.x, rename the

schema files to *.xml*, or define a "helper application" within Netscape by choosing **Edit/Preferences > Navigator/Helper Applications > New Type**. Select the following options: **Extension**: tsd **Mime Type**: text/tsd **Handled by**: (blank).

Load XML Documents into a Database

Once you have defined the schema, you can load documents into your database. In the Tamino Interactive Interface, choose the form **Load contents of a file to insert or update documents** by selecting the **Load** tab. A screen similar to the following is displayed:



Input for the field **Database URL** is taken from the specifications you made before. If you want to switch to a different database, enter a new URL (see ***Database URL***). The input for the field **Encoding** is the browser's default value (see ***Encoding***).

To load XML data into the specified database, go to the field **Load file**. In this field you specify the name of a file that contains the XML object to be loaded into the database. You can either type in the filename and full path name directly, or you can use the **Browse** button that is located at the end of the field to locate the file that you want.

The following example uses the sample data from the XML instances *bloggs.xml* and *atkins.xml* supplied with the Tamino Documentation. The data are in the directory *<TaminoDocRootDir>/examples/patient*, where *<TaminoDocRootDir>* is the starting directory of the product documentation.

Now load the XML instance *bloggs.xml* into the database. Before loading instances into the database, ensure that the corresponding schema is defined in the database (see section ***Define or Update a Schema***). Select the XML instance *bloggs.xml*, using the **Browse** button. Also, specify the collection "Hospital" in the field **Into collection**. Finally, choose the **Load** button when you have specified all input.

If the load is successful, you will get a response document containing lines like the following in the result window (works on XML-capable browsers only!):

```
<?xml version="1.0" encoding="windows-1252" ?>
- <ino:response xmlns:ino="http://namespaces.softwareag.com/tamino/response2"
    xmlns:xql="http://metalab.unc.edu/xql/">
  - <ino:message ino:returnvalue="0">
      <ino:messageline>document processing started</ino:messageline>
    </ino:message>
    <ino:object ino:collection="Hospital" ino:doctype="patient" ino:id="1" />
  - <ino:message ino:returnvalue="0">
      <ino:messageline>document processing ended</ino:messageline>
    </ino:message>
  </ino:response>
```

If the XML object was loaded successfully, you will see in two `ino:message` elements of this response document that the return value (`ino:returnvalue`) is zero, which indicates successful completion.

In the `ino:object` element you see the name of the doctype and the collection into which the object was loaded. In this example, the doctype name is `patient` and the collection name is `Hospital`.

You also see that the object ID of the object you loaded has been assigned the internal ID of “1” (`ino:id="1"`). When you are retrieving objects at a later stage using database queries, you can use this object ID to retrieve a particular record.

Here is another sample XML object that you can load by the same method. Browse to the sample XML file *atkins.xml* and choose the **Load** button. The object is loaded into the database.

You should get a response document that is identical to the response document for the first object apart from one difference, namely that the object ID has a different value.

```
<?xml version="1.0" encoding="windows-1252" ?>
- <ino:response xmlns:ino="http://namespaces.softwareag.com/tamino/response2"
    xmlns:xql="http://metalab.unc.edu/xql/">
  - <ino:message ino:returnvalue="0">
      <ino:messageline>document processing started</ino:messageline>
    </ino:message>
    <ino:object ino:collection="Hospital" ino:doctype="patient" ino:id="2" />
  - <ino:message ino:returnvalue="0">
      <ino:messageline>document processing ended</ino:messageline>
    </ino:message>
  </ino:response>
```

If you plan to load large amounts of data into Tamino, it is generally better to use one of Tamino's data loading utilities, especially the Tamino Data Loader. Refer to the documentation on the *Tamino Data Loaders* for more details.

For testing purposes, create and load a few more XML objects with the same structure as the first two. You can then use these for performing queries or for deleting objects.

Load Non-XML Data into the Database

If you want to load documents that do not have an XML format, for example graphic or word processing files, a special schema is needed. Here is an example of a schema file for non-XML data:

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
   xmlns:xs  = "http://www.w3.org/2001/XMLSchema"
   xmlns:tsd = "http://namespaces.softwareag.com/tamino/TaminoSchemaDefinition"
   >
   <xs:annotation>
      <xs:appinfo>
         <tsd:schemaInfo name = "abcNonXML">
            <tsd:collection name = "abcNonXML"/>
            <tsd:doctype name = "xyzNonXML">
                <tsd:nonXML/>
            </tsd:doctype>
         </tsd:schemaInfo>
      </xs:appinfo>
   </xs:annotation>
</xs:schema
```

The decisive element in this schema is `tsd:nonXML`. It tells Tamino to not treat the data as XML data.

To load the data into Tamino, first define the schema above to Tamino. The next step is to load the non-XML files into Tamino.

Use the Interactive Interface as follows:

To load non-XML data into Tamino with the Tamino Interactive Interface

1 Choose the **Load** tab.

2 Enter the database URL.

3 Enter the file to be loaded. Use the **Browse** button to locate the file, if necessary.

4 The entry in the field **Into collection** is special for non-XML data. Enter the following:.

```
(collection name)/(doctype name)/(document name)
```

If, for example, you want to load a file named *patient.doc* with the example schema file above, enter:

```
abcNonXML/xyzNonXML/patient.doc
```

The document with ino:docname *patient.doc* is loaded into Tamino, and you can query for it.

Specifying the document name is optional, but recommended, since it provides the possibility to query the document via the ino:docname attribute (with XQuery). Use the function `tf:getDocname` to get the document name for the non-XML document element.

To query non-XML data with the Tamino Interactive Interface

1 Choose the **X-Query** tab.

2 Enter the database URL.

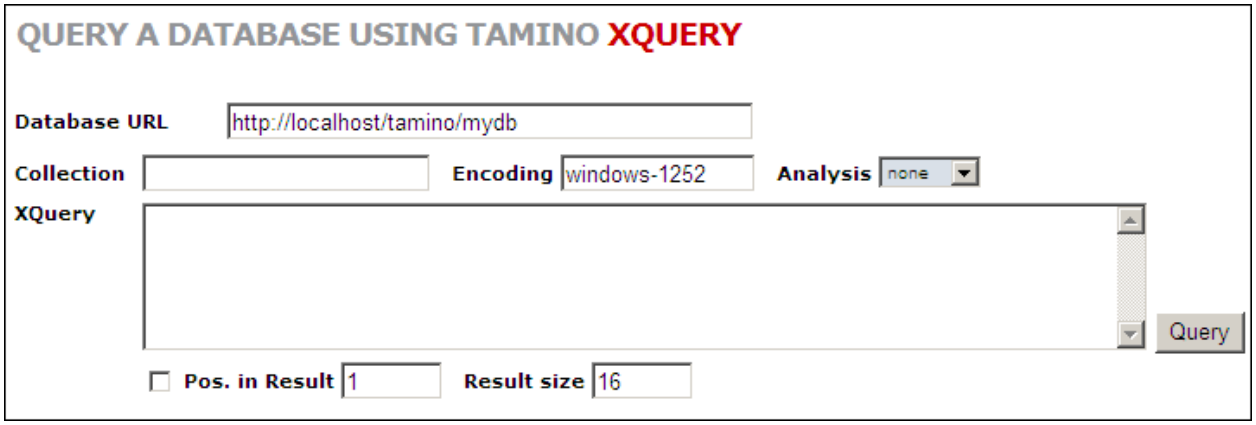
3 In the **Collection** field, enter the following:

```
(collection name)/(doctype name)/(document name)
```

You do not need to enter anything in the **X-Query** field, as any input in this field is ignored. Note that it is not possible to use the **XQuery** form for querying non-XML data, as plain URL addressing is necessary. With the Interactive Interface, plain URL addressing can only be done with the **X-Query** form.

Query a Database with Tamino XQuery

Tamino supports two query languages, the Tamino specific X-Query (spoken X-dash-Query) and the new Tamino XQuery (without a "dash"), which follows the proposals of the W3C about XML query languages. If you want to use the new Tamino XQuery (without the dash), choose the **Query a database using XQuery** form by selecting the appropriate tab in the Tamino Interactive Interface. A screen similar to the following is displayed:



Input for the field **Database URL** is taken from the specifications you made before. If you want to switch to a different database, enter a new URL (see *Database URL*). The input for the field **Encoding** is the browser's default value (see *Encoding*).

For querying the data, you need to specify the collection into which the objects belong. Enter the name of the collection that you wish to access. For this example, the name of the collection is "Hospital".

If you do not want to display the complete result set of your query, you can specify the index of the first document to be returned in the field **Position in Result**.

If you want to include an analysis of the query, choose the option **Default** in the **Analysis** list box. Otherwise, the query is performed without analysis. For further information, see the **reference section** about XQuery.

Also, it is possible to limit the number of documents returned. Enter a number in the field **Result Size**. It is also possible to enter the value "0". This is necessary for update queries.

In the field **XQuery**, you can enter queries to the selected database and collection using the new Tamino XQuery language according to proposed W3C standards. Refer to XQuery 4 User Guide for more information about this query language.

The following examples assume that you have loaded the two XML objects for Peter Atkins and Fred Bloggs, as described in the section *Load XML Documents into a Database* into a test database and collection.

In the **XQuery** field, enter the following query, then choose the **XQuery** button.

```
input()/patient
```

This query means: find all elements of type `patient` in the current collection (Hospital).

You should receive a response document that contains lines similar to these (other header and general processing information is also returned, as well as more data about the patients Atkins and Bloggs, but not shown in the examples):

```
- <xq:result xmlns:xq="http://namespaces.softwareag.com/tamino/XQuery/result"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <patient>
    - <name>
        <surname>Bloggs</surname>
        <firstname>Fred</firstname>
      </name>
      <sex>Male</sex>
      <born>1950</born>
    - <address>
        <street>Mill Lane</street>
        <housenumber>16</housenumber>
        <city>Bradford</city>
        <postcode>BRAML0225</postcode>
      </address>
```

Now change the query to

```
input()/patient/name/surname
```

and choose the **Query** button. The response document should contain the following lines:

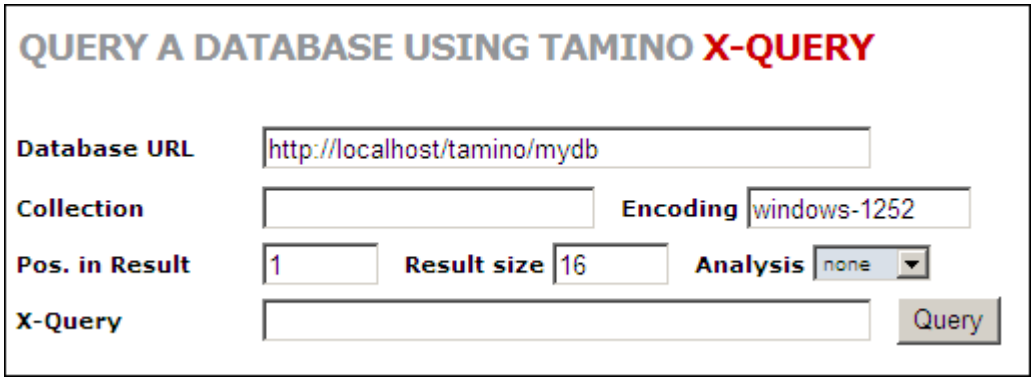
```
- <xq:result xmlns:xq="http://namespaces.softwareag.com/tamino/XQuery/result"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <surname>Atkins</surname>
    <surname>Bloggs</surname>
  </xq:result>
```

Note that this time, only the text in the `surname` element has been returned.

For a description of the XQuery language, refer to the document XQuery 4 User Guide or the XQuery 4 Reference Guide.

Query a Database with Tamino X-Query

Tamino supports two query languages, the Tamino specific X-Query (spoken “X-dash-Query”) and Tamino XQuery (without a “dash”), which follows the recommendation of the W3C about XML query languages. If you want to use the Tamino specific query language X-Query, choose the **Query a database using Tamino X-Query** form by selecting the appropriate tab in the Tamino Interactive Interface. A screen similar to the following is displayed:



Input for the field **Database URL** is taken from the specifications you made before. If you want to switch to a different database, enter a new URL (see *Database URL*). The input for the field **Encoding** is the browser's default value (see *Encoding*).

For querying the data, you need to specify the collection to which the objects belong. Enter the name of the collection that you wish to access. For this example, the name of the collection is "Hospital".

If you do not want to display the complete result set of your query, you can specify the index of the first document to be returned in the field **Position in Result**.

Also, it is possible to limit the number of documents returned. Enter a number in the field **Result Size**. If you enter the value "0", cursoring is switched off.

Note: If you switch off cursoring, there is no limit for the result size. Thus the result size may become very large.

In the field **X-Query**, you can enter queries to the selected database and collection using the Tamino XML database query language X-Query. Refer to the X-Query User Guide for more information about this query language.

The following examples assume that you have loaded the two XML objects for Peter Atkins and Fred Bloggs, as described in the section *Load XML Documents into a Database* into a test database and collection.

In the **X-Query** field, enter the following query, then choose the **Query** button.

```
patient
```

This query means: find all elements of type `patient` in the current collection (Hospital).

You should receive a response document that contains lines similar to these (other header and general processing information is also returned, as well as more data about the patients Atkins and Bloggs, but not shown in the examples):

```
- <xql:result>
  - <patient ino:id="1">
    - <name>
        <surname>Bloggs</surname>
        <firstname>Fred</firstname>
      </name>
      <sex>Male</sex>
      <born>1950</born>
    - <address>
        <street>Mill Lane</street>
        <housenumber>16</housenumber>
        <city>Bradford</city>
        <postcode>BRAML0225</postcode>
      </address>
```

Now change the query to

```
patient/name/surname
```

and choose the **Query** button. The response document should contain the following lines.

```
- <xql:result>
    <surname ino:id="1">Bloggs</surname>
    <surname ino:id="2">Atkins</surname>
  </xql:result>
```

Note that this time, only the `surname` element has been returned.

Analysis: Request information concerning the query

The Tamino Interactive Interface offers the possibility to obtain information about how the query is processed. This may be helpful for performance reasons, e.g. if you want to find out if the post processor is used for the query and if there might be another possibility of issuing a query request. See the documentation of ino:explain for detailed information.

To obtain information concerning the query

1 Enable the **Analysis** dropdown box by selecting one of the possible values in the list. You can choose among the following options:

Option	Description
None	No analysis is done, the query is executed.
Default	Default analysis of the query is done, using the function `ino:explain`.
Path	Analysis is done with explanation level `path` using `ino:explain`.
Tree	Analysis is done with explanation level `tree` using `ino:explain`.

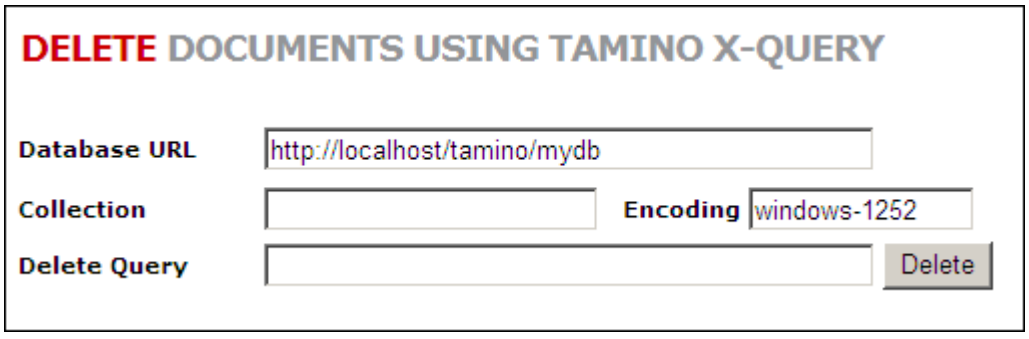
2 Enter a query into the **X-Query** field.

3 Execute the query by choosing the **X-Query** button.

In the result window, information about the query is displayed.

Delete XML Objects from a Database

Using this command, you can delete XML objects that you have previously loaded. In the Tamino Interactive Interface, choose the **Delete documents using Tamino X-Query** form by selecting the **Delete** tab. A screen similar to the following is displayed:



Use X-Query syntax to identify one or more objects on the basis of their content.

To select an object for deletion based on its content, proceed as follows. If, for example, you want to delete the object for the patient named Fred Bloggs, enter the collection name "Hospital" and then the following command in the **Delete Query** field: "patient[name/surname="Bloggs"]".

You can use wildcards to select multiple objects for deletion. If, for example, you type "patient[name/surname~="B*"]"

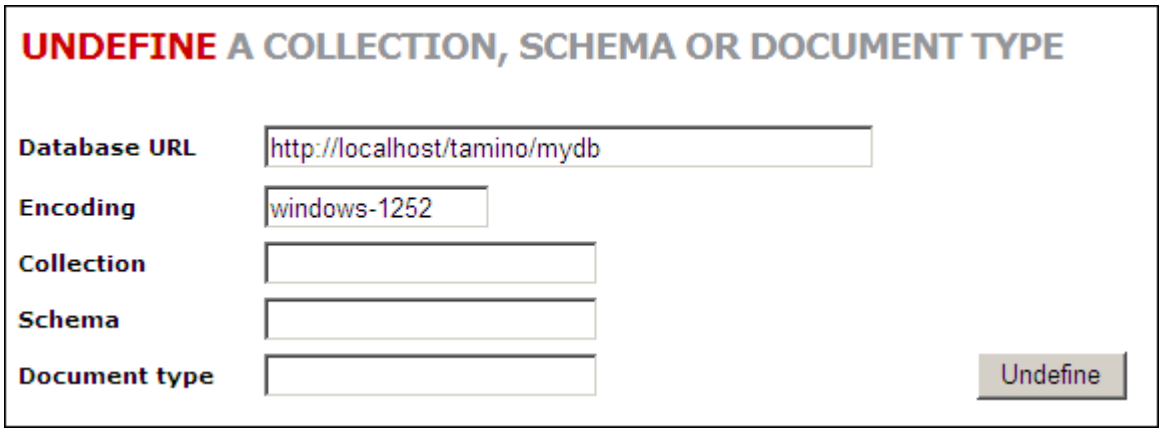
you will select for deletion all objects in the schema "patient" in which the surname begins with the letter "B".

To select an object for deletion based on its object ID, proceed as follows. If, for example, you want to delete the object for the patient named Fred Bloggs, which was created with an `ino:id` of "1", enter the following command: "patient[@ino:id=1]" and choose the **Delete** button.

If the command is successful, the response document will inform you that “document(s) have been deleted”.

Undefine a Collection, Schema, or Doctype

If you do not need a schema, collection or doctype any longer, the Tamino Interactive Interface offers a convenient way of undefining one or the other or all of them from the Tamino database that you have specified. Choose the form **Undefine a collection, schema or document type** by selecting the appropriate tab in the Tamino Interactive Interface. A screen similar to the following is displayed:



Here you specify the name of a collection, schema or document type that you want to undefine for the current database.

Caution: If you undefine a *doctype*, its data is deleted and it is removed from its enclosing schema. Undefining a *schema* will remove the schema and all doctypes including their data,

that are part of the schema. Consequently, undefining a *collection* will remove the collection with all schemas and doctypes and all documents.

If you follow the example given in the section ***Define: Define a collection or schema***, you have a collection called "Hospital" that contains a doctype called "patient" in the database "mydb".

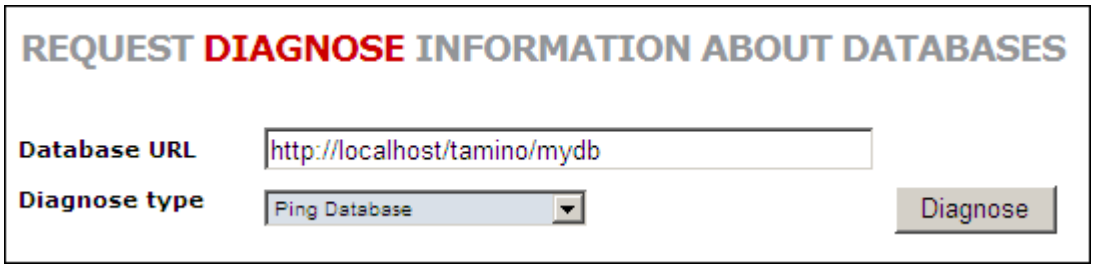
To undefine the doctype "patient" from the collection "Hospital", enter the names in the appropriate fields **Collection**, **Schema** and **Document type** , then select the **Undefine** button. Make sure that you have entered the correct database URL and schema file.

If the command is successful, the response document will contain a message that the doctype and collection have been undefined. Also, the data for the patient Atkins have been deleted from the database along with all other documents in that doctype.

Note that the collection name, schema name and doctype name are case-sensitive, i.e. "hospital" is not the same as "Hospital", and "Patient" is not the same as "patient".

Request Diagnose Information about the Database

The Tamino Interactive Interface provides the possibility to send diagnostic requests to a Tamino database. Choose the form **Request diagnose information about databases** by selecting the appropriate tab in the Tamino Interactive Interface. A screen similar to the following is displayed:



Choose a diagnose type for the current database. The following types are available:

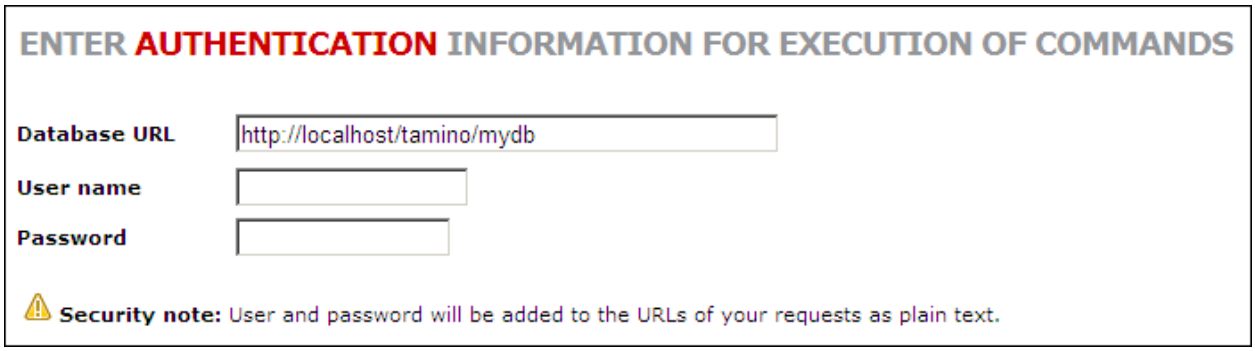
Diagnose Type	Description
Ping Database	Tries to establish an HTTP connection and returns a positive answer in the case that the server could be reached and is online.
Show Database Version	Returns the version of the Tamino Server.
Request Echo from Database	Delivers the HTTP headers seen in the Tamino Server if X-Machine is reachable by the web server, otherwise an error message is generated by the web server.
Get Time Information	Returns the total amount of time that the server has been active in user mode (i.e. executing user requests) since the database server was started, and the total amount of time that the server has been active in system (kernel) mode (i.e. executing system calls that result from user requests) since the database

Diagnose Type	Description
	server was started. The values returned are the total times for all users together, not the individual times for each user.

When you have selected the diagnose type, choose the button **Diagnose**. Diagnose information about the current database is displayed in the result window. For further information, see the documentation about X-Machine Programming.

Enter Authentication Information

If a Tamino database or collection is protected and can be modified only by certain users, a user name and password have to be provided to work with the database. Choose the form **Enter authentication information for execution of commands** by selecting the appropriate tab in the Tamino Interactive Interface. A screen similar to the following is displayed:



Enter your user name and password for the current database in the appropriate fields. The information is used on all other forms of the Tamino Interactive Interface.

Caution: User name and password are added to the URLs of your request as plain text. You might want to switch to secure communication (https) if this causes problems in your environment.

5 Reference: XQuery

Tamino supports two query languages, the Tamino specific X-Query (spoken X-dash-Query) and the new Tamino XQuery (without a "dash"), which follows the proposals of the W3C about XML query languages. XQuery is the W3C query language supported by Tamino to perform queries on XML objects and on non-XML objects. The form allows you to send XQuery commands to Tamino in order to retrieve database content.

See also: **Query a Database with XQuery**. For detailed information about XQuery, see XQuery 4 User Guide or the XQuery 4 Reference Guide.

The following elements are available on this form:

Element	Description	Corresponding URL Parameter
Database URL	Specifies the database. See **Database URL**	`databaseurl`
Favorite	Saves the settings as favorite (only Microsoft Internet Explorer 5 or higher). See **Favorite**	-
Collection	Specifies the collection to use. See **Collection**	`collectionxmlq`
Encoding	Specifies the character encoding. See **Encoding**	`encodingxmlq`
Pos. in Result	Specifies the index of the first document to be returned.	`positionxmlq`
Result size	Specifies the number of documents to be returned. Value needs to be 0 for XQuery update operations (insert, delete, rename, replace).	`resultsizexmlq`
Analysis	Analyzes the query. Possible values are: ■ `none`: No analysis ■ `Default`: Default analysis of the query using the query processing instruction `{?explain?}`	`explainxq`
XQuery	Queries the database with W3C XQuery syntax. Allows to retrieve and update documents.	`xmlquery`
Query	Button to submit the query.	`-runcmd=xquery`

Element	Description	Corresponding URL Parameter
Help	Displays online documentation about this form.	-

6 Reference: X-Query

Tamino supports two query languages, the Tamino specific X-Query (spoken X-dash-Query) and the new Tamino XQuery (without a "dash"), which follows the proposals of the W3C about XML query languages. X-Query is the XML query language used in Tamino to perform queries on XML objects and on non-XML objects. The form allows you to send X-Query commands to Tamino in order to retrieve database content.

See also: **Query a Database with Tamino X-Query**. For detailed information about X-Query, see the X-Query User Guide or the X-Query Reference Guide.

The following elements are available on this form:

Element	Description	Corresponding URL Parameter
Database URL	Specifies the database. See **Database URL**	`databaseurl`
Favorite	Saves the settings as favorite (only Microsoft Internet Explorer 5 or higher). See **Favorite**	-
Collection	Specifies the collection to use. See **Collection**	`collection`
Encoding	Specifies the character encoding. See **Encoding**	`encoding`
Pos. in Result	Specifies the index of the first document to be returned.	`position`
Result size	Specifies the number of documents to be returned. If value is 0, cursoring is switched off. In this case, result size can be very high.	`resultsize`
Analysis	Analyzes the query. Possible values are: ■ `none`: No analysis; execute query ■ `Default`: Default analysis of the query using the function `ino:explain(QueryText)` ■ `Path`: Receiving explanation level "path" using `ino:explain(QueryText,path)`	`explain` Possible values: ■ false (= none) ■ true (= default) ■ path (= path)

Element	Description	Corresponding URL Parameter
	■ `Tree`: Receiving explanation level “tree” using `ino:explain(QueryText,tree)`	■ tree (= tree)
X-Query	Queries the database with Tamino X-Query syntax.	`query`
Query	Button to submit the X-Query.	`runcmd=query`
Help	Displays online documentation about this form.	-

7 Reference: Delete

This form allows you to delete XML objects from the current database that you have previously loaded.

See also: **Delete XML Objects from a Database**.

The following elements are available on this form:

Element	Description	Corresponding URL Parameter
Database URL	Specifies the database. See **Database URL**	`databaseurl`
Favorite	Saves the settings as favorite (only Microsoft Internet Explorer 5 or higher). See **Favorite**	-
Collection	Specifies the collection to use. See **Collection**	`collectiondel`
Encoding	Specifies the character encoding. See **Encoding**	`encodingdel`
Delete Query	Specifies the query for the instances to be deleted.	`delete`
Delete	Button to submit the deletion	`runcmd=delete`
Help	Displays online documentation about this form.	-

8 Reference: Load

This form allows you to load XML objects into the database.

See also: **Load XML Documents into a Database**.

The following elements are available on this form:

Element	Description	Corresponding URL Parameter
Database URL	Specifies the database. See **Database URL**.	`databaseurl`
Collection	Specifies the collection to use. See **Collection**.	`collectionload`
Encoding	Specifies the character encoding. See **Encoding**.	`encodingload`
Load file	Path of the file to be loaded. If it is not an XML file, but e.g. a text file, the encoding field will be used by Tamino to interpret the contents.	-
Into collection	Specifies the name of the target collection.	`collectionload`
Load	Button to submit the load request.	-
Help	Displays online documentation about this form.	-

9 Reference: Define

This form allows you to define or update a collection or schema in a Tamino database.

See also: **Define or Update a Schema**.

The following elements are available on this form:

Element	Description	Corresponding URL Parameter
Database URL	Specifies the database. See **Database URL**.	`databaseurl`
Schema file	Specifies the file that contains the schema definition.	-
Define with instance validation	Specifies the mode of the schema definition. If you check this box, the schema is defined *and* validated.	`_mode="validate"`
Define	Button to submit the define command	
Help	Displays online documentation about this form.	-

10 Reference: Undefine

This form allows you to remove a document type, schema or collection from a Tamino database.

See also: **Undefine a Collection, Schema, or Doctype**.

The following elements are available on this form:

Element	Description	Corresponding URL Parameter
Database URL	Specifies the database. See **Database URL**.	`databaseurl`
Favorite	Saves the settings as favorite (only Microsoft Internet Explorer 5 or higher). See **Favorite**.	-
Encoding	Specifies the character encoding. See **Encoding**.	`encodingundef`
Collection	Specifies the name of the collection (mandatory). If schema and document type are left empty, the collection with all schemas and document types will be deleted. See also **Collection**.	`encodingundef`
Schema	Specifies the name of the schema (optional, if no document type is entered). If a schema is entered, but no document type, the schema residing in the collection will be undefined, including all document types defined in this schema.	`schemaundef`
Document type	Specifies the name of the document type (optional). If entered, the document type residing in the collection and schema will be undefined.	`undefine`
Undefine	Button to submit the undefine command.	`runcmd=undefine`
Help	Displays online documentation about this form.	-

11 Reference: Diagnose

This form allows you to send diagnose requests to a Tamino database.

See also: **Request Diagnose Information about the Database**.

The following elements are available on this form:

Element	Description	Corresponding URL Parameter
Database URL	Specifies the database. See **Database URL**	`databaseurl`
Favorite	Saves the settings as favorite (only Microsoft Internet Explorer 5 or higher). See **Favorite**	-
Diagnose type	Displays a list with possible diagnose parameters to be sent to a Tamino database.	`diagnose`
Diagnose	Button to submit the diagnose request to Tamino.	`runcmd=diagnose`
Help	Displays online documentation about this form.	-

12 Reference: Authentication

This form allows you to enter a user name and password to be used in requests sent to a Tamino database. When using the Tamino Interactive Interface with a database where authentication is required, you can send requests only if this authentication information is provided.

See also: **Enter Authentication Information**.

The following elements are available on this form:

Element	Description	Corresponding URL Parameter
Database URL	Specifies the database. See **Database URL**	`databaseurl`
User name	Name of the user registered in the Tamino database.	`dbuser`
Password	The user's password.	-
Help	Displays online documentation about this form.	-

Index