software AG

Tamino

Tamino Non-XML Indexer

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: XNE-DOC-97-20160318

Table of Contents

Tamino Non-XML Indexer

This document provides information for using the Tamino Non-XML Indexer.

The information is in the form of narrative descriptions and reference information that lists the MIME types (Multipurpose Internet Mail Extensions types) that are supported “out of the box”, how to extend the software to support other MIME types, and the metadata that may be associated with each MIME type. The documentation is intended for software developers who are writing applications that use non-XML files stored in a Tamino database.

You should be familiar with the basic principles of Tamino, the Tamino Manager, and X-Tension: Tamino Server Extensions. For further information on these topics, see the relevant Tamino XML Server Documentation.

I Introduction

1 Introduction

The Tamino Non-XML Indexer seamlessly integrates non-XML files, for example Star Office documents and Microsoft Office documents, into your Tamino environment. You can now make meaningful searches on the content and/or metadata of legacy non-XML files. The metadata can typically include information such as the date when the document was last changed, the author, etc. Note that the amount and type of metadata depends on the application program that created the file. Older versions of software, for example Microsoft Word Version 2.0, often generate little or no metadata.

When a non-XML file is processed (stored or updated) in a Tamino database collection in which the Tamino Non-XML Indexer is active, Tamino stores two objects:

- the non-XML file itself;
- a so-called shadow file, which is indexed XML data comprising:
  - the raw data contained in the file, for example the plain text in a Microsoft Word file;
  - metadata extracted from the file.

Note that it is possible to suppress the storing of the non-XML file; this is meaningful if, for example, it is already stored elsewhere. In order to do this, use specify the element `tsd:storeShadowOnly` in the document's schema. When this option is active, a pseudo non-XML file which is a BLOB of size zero is stored.

The Tamino Non-XML Indexer processes each document based on its MIME type; this information is submitted along with the document when the document is stored in Tamino. The list of MIME types that are supported by the Tamino Non-XML Indexer “out of the box” is in the chapter **Supported MIME Types**. You can add support for further MIME types by following the instructions in the section **Adding Support for Further MIME Types**.The following is an informal, incomplete list of applications that produce documents that can be processed by the Tamino Non-XML Indexer. Some MIME types, for example "text/rtf", are generic, i.e. files with these MIME types can be produced by many different applications, including freeware and shareware programs.

- Microsoft Office files:
  - Microsoft Word
  - Microsoft Excel
- OpenOffice files:
  - OpenOffice Writer
  - OpenOffice Calc
- StarOffice files:
  - StarOffice Writer
  - StarOffice Calc
- Adobe PDF files
- Plain text files (UTF-8)
- Plain text files
- MPEG audio files (often known as MP3 files)
- RTF (Rich Text Format) files
- Zip files

II Component Profile and Set-Up

2 Component Profile and Set-Up

This chapter lists the prerequisites and procedures for successfully using the Tamino Non-XML Indexer.

Component Profile

Here you will find general information about the Tamino Non-XML Indexer and how to deploy it.

Tamino Component Profile for the Tamino Non-XML Indexer	
Required Software	**Software AG Products:** Software AG Tamino XML Server **Third-Party Products:** Apache POI JDOM Apache Xerces Apache Log4j
Location of Installed Component	Under Linux and Solaris: *$SAG/ino/vnnnn/ServerExtensions/NonXMLIndexer* (henceforth called *TaminoXNEDir*). Under Microsoft Windows: *...\Software AG\Tamino\Tamino n.n.n.n\ServerExtensions\NonXMLIndexer* (henceforth called *TaminoXNEDir*)
Component Files	**Schema files:** *TaminoXNEDir\etc\dc.tsd* *TaminoXNEDir\etc\meta.tsd* *TaminoXNEDir\etc\office.tsd* *TaminoXNEDir\etc\template.tsd* **Server extension software:** *TaminoXNEDir\sxs\nixe.sxp*
Bundled Software	none

Third-Party Products

The following third-party products are used by the Tamino Non-XML Indexer:

- JDOM

  For more information, see **http://www.jdom.org/**
- Log4j from the Apache Logging Project

For more information, see **http://logging.apache.org/**

- Apache POI

  For more information, see **http://poi.apache.org/**

- Apache Xerces (Xerces2 Java parser)

  For more information, see **http://xerces.apache.org/**

Set-Up

Please refer to the section **Setting Up the Tamino Non-XML Indexer**.

III Architectural Overview

3 Architectural Overview

The Tamino Non-XML Indexer is based on and extends Tamino's facilities for storing and retrieving non-XML objects; these facilities are described in the sections Storing Non-XML Objects in Tamino, Using Shadow Functions and Shadow Functions. Tamino's basic functionality allows you to store, update and delete non-XML objects; you can retrieve a non-XML object on the basis of its ino:id or its ino:docname. These two pieces of information, the ino:id and the ino:docname, can be regarded as metadata associated with the original non-XML object.

The Tamino Non-XML Indexer plugin modules, which can be either the standard plugins supplied by Software AG or user-written plugins, work in conjunction with Tamino to extend the set of metadata that is associated with non-XML objects. The plugin produces a well-formed XML document that conforms to the metadata schema. Typically, the plugin populates the contents of the elements and the attributes with values contained in or derived from the original non-XML file, for example author's name, date last modified, or file size. For certain kinds of input data, the plugin can derive data directly from the input data; for example, the PDF plugin can read the text contents out of most PDF files and store them in the shadow file.

Whenever an insert or update operation is processed for a document whose MIME type has been declared to the Tamino Non-XML Indexer, Tamino calls the server extension (`SXSBlobIndexer`). The server extension's dispatcher then activates the appropriate method in the appropriate plugin. The plugin:

1. Analyses the document, and extracts the raw data contained in the document if appropriate (for example, the plain text is extracted from a Microsoft Word document) and also metadata that describes the document;
2. Creates a well-formed XML document containing this information;
3. Passes this XML document as a shadow document to Tamino, which writes it to the database.

Both the non-XML object (or the pseudo non-XML object, which is a BLOB of size zero, if the option "storeShadowOnly" is activated) and its associated XML shadow object have the same ino:id.

If the document schema includes the tsd:noConversion element, the document is processed as follows:

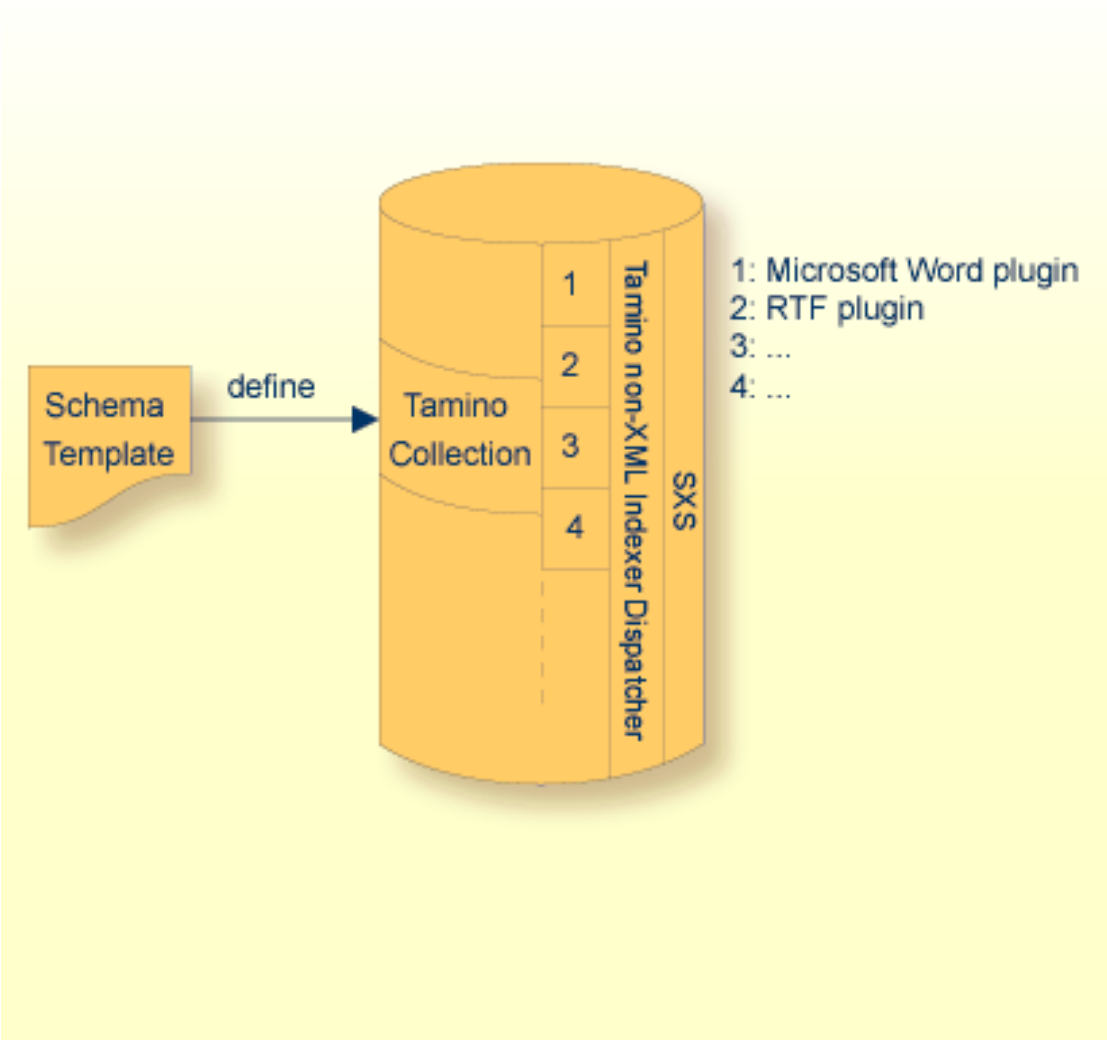
The Tamino Non-XML Indexer receives the document as an unaltered bytestream. (In particular, the document has not yet been subjected to Unicode translations.)

If the MIME-type indicates that the document is a text file, then:

- If there is no encoding information, the shadow document is empty and a warning is issued;
- If translation is not supported, the shadow document is empty and a warning is issued;
- Otherwise, the Tamino Non-XML Indexer translates the incoming bytestream into a text stream as indicated by the encoding information.

If translation succeeds, the Tamino Non-XML Indexer processes the resulting string as though the string had been passed to the putText() function.

The non-XML object can be accessed by plain URL addressing, unless the option "storeShadowOnly" has been activated. The associated XML shadow document can be accessed by issuing a query (for example in XQuery or X-Query form). Note that non-XML objects stored in Tamino can be accessed in this manner even if the Tamino Non-XML Indexer is not installed; the difference is in the variety of different metadata fields that can be queried.



Tamino Architecture

Tamino Non-XML Indexer services are logically provided by software at two levels, as shown in the diagram above:

1. The Tamino Non-XML Indexer dispatcher;
2. One or more plugin modules (represented in the diagram by the numbers 1, 2, 3, 4...). Each plugin services one or more MIME types.

IV Setting Up the Tamino Non-XML Indexer

4 Setting Up the Tamino Non-XML Indexer

The procedures for installing a Tamino Server Extension are detailed in the Tamino documentation. Please see the section *Installing a Tamino Server Extension*.

The installation is done in three distinct phases:

1. On each Tamino server (computer) where Tamino Non-XML Indexer services are required, **install the Tamino Non-XML Indexer software**.
2. For each Tamino database where Tamino Non-XML Indexer services are required, **configure SXS**.
3. For each doctype where Tamino Non-XML Indexer services are required, **define an appropriate schema**.

Installing the Tamino Non-XML Indexer Software

Installing under Linux and Solaris

The Tamino Non-XML Indexer is automatically installed along with the Tamino XML Server.

Continue with the steps described in the section **Configuring the Tamino Non-XML Indexer**.

Installing under Microsoft Windows

The Tamino Non-XML Indexer is automatically installed if you select the "Complete" installation option of the Tamino XML Server. Continue with the steps described in the section **Configuring the Tamino Non-XML Indexer**.

Configuring the Tamino Non-XML Indexer

Using the Tamino Manager:

1. Select your Tamino database, then select **Server Extensions**, then choose the command **Install Extension**.
2. Browse to the directory *TaminoXNEDir*/*sxs*. Select the file *nixe.sxp*.

Notes:

a. For UNIX systems, *TaminoXNEDir* is by default *$SAG*/*Tamino*/*Tamino n.n.n.n*/*ServerExtensions*/*NonXMLIndexer*.

b. For Microsoft Windows systems, *TaminoXNEDir* is by default ...*Software AG**Tamino**Tamino n.n.n.n**ServerExtensions**NonXMLIndexer*.

3. Enter the required user-ID.
4. Choose **OK** to install the Tamino Non-XML Indexer.
5. Select your Tamino database, then select **Server Extensions**, then select **SXSBlobIndexer**. Choose the command **Modify Extension**.
6. Modify the classpath to include *TaminoXNEDir*/*lib*/**.jar*.

For more information about the Tamino Manager, please see the documentation *Tamino Manager*.

Modifying and Installing the Schema Template

As an added convenience for our customers, the Tamino Non-XML Indexer installation kit includes a schema template. To use this template, perform the following steps:

1. Copy the schema template files:

   *TaminoXNEDir**etc**dc.tsd*
   *TaminoXNEDir**etc**meta.tsd*
   *TaminoXNEDir**etc**office.tsd*
   *TaminoXNEDir**etc**template.tsd*

   to a working directory.
2. Edit each of these four files; in each file, replace the string *<my_collection>* by the name of your collection.

   Note: The 15-character string that you must replace is *<my_collection>*, including the angle brackets. The replacement string must not include angle brackets. If it includes angle brackets, or if you forget to perform the replacement, the schema will be syntactically invalid.

3. In the file *template.tsd*:
   - Replace the string *<my_schema>* by the name of your schema;
   - Replace each occurrence of the string *<my_doctype>* by the name of your doctype.
4. Install the four schema files in your Tamino collection, in the sequence:
   a. *dc.tsd*
   b. *meta.tsd*
   c. *office.tsd*
   d. *template.tsd*

   You can use the Tamino Schema Editor or the Tamino Interactive Interface to do this. (For more information, see the section *Tamino Interactive Interface*.)

If you wish to use the Tamino Non-XML Indexer with more than one collection, repeat steps 2, 3 and 4 as necessary.

The installation of the Tamino Non-XML Indexer is now complete.

V Using the Tamino Non-XML Indexer

5 Using the Tamino Non-XML Indexer

After the steps listed in the section **Setting Up the Tamino Non-XML Indexer** have been successfully completed, the product is ready for use. You can now make meaningful searches on the content and/or metadata of legacy non-XML files.

The typical sequence of steps when using the Tamino Non-XML Indexer is as follows:

1. Store a non-XML document (this explanation assumes that the document is a PDF file).

   The Tamino Non-XML Indexer intercepts the store operation; using the file's MIME type (application/pdf), the dispatcher activates the PDF plugin. The PDF plugin reads the PDF file and creates an XML shadow file containing:

   - metadata that describes the PDF file: date, title, author, creator and producer;
   - the text contents of the PDF file.

   The Tamino Non-XML Indexer passes the shadow file to Tamino, which stores it in the database.

   The intercepted original document (the PDF file in our example) is also passed back to Tamino, which stores it in the database, unless the option "storeShadowOnly" is activated.

2. You can now issue queries, for example to retrieve PDF files with author="John Smith" or to retrieve PDF files that contain the text string "the importance of being earnest".

   Note that the type of queries that you can issue depends on the contents of the shadow file. See the section **Mapped Properties** or the documentation of user-written plugins for more details.

VI Supported MIME Types

6 Supported MIME Types

The Tamino Non-XML Indexer is currently delivered with plugins that support the following MIME types. Note, however, that the contents of OLE objects embedded in Microsoft Office files cannot be indexed. If you need to support other MIME types, see the chapter **Adding Support for Further MIME Types**.

MIME Type	Application
application/ms-excel	Microsoft Excel
application/msexcel	Microsoft Excel
application/msword	Microsoft Word
application/pdf	Adobe Portable Document Format (P
application/rtf	Microsoft Word
application/vnd.ms-excel	Microsoft Excel
application/vnd.ms-excel.addin.macroEnabled.12	Office Open XML: xlam
application/vnd.ms-excel.sheet.macroEnabled.12	Office Open XML: xlsm
application/vnd.ms-powerpoint.addin.macroEnabled.12	Office Open XML: ppam
application/vnd.ms-powerpoint.presentation.macroEnabled.12	Office Open XML: pptm
application/vnd.ms-powerpoint.slideshow.macroEnabled12	Office Open XML: ppsm
application/vnd.ms-word.document.macroEnabled.12	Office Open XML: docm
application/vnd.openxmlformats-officedocument.presentationml.presentation	Office Open XML: pptx
application/vnd.openxmlformats-officedocument.presentationml.slideshow	Office Open XML: ppsx
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	Office Open XML: xlsx
application/vnd.openxmlformats-officedocument.wordprocessingml.document	Office Open XML: docx
application/vnd.sun.xml.calc	StarOffice Calc
application/vnd.sun.xml.calc.template	StarOffice Calc
application/vnd.sun.xml.draw	StarOffice Draw
application/vnd.sun.xml.draw.template	StarOffice Draw

MIME Type	Application
application/vnd.sun.xml.impress	StarOffice Impress
application/vnd.sun.xml.impress.template	StarOffice Impress
application/vnd.sun.xml.math	
application/vnd.sun.xml.writer	StarOffice Writer
application/vnd.sun.xml.writer.global	StarOffice Writer
application/vnd.sun.xml.writer.template	StarOffice Writer
application/x-zip-compressed	zip archive
application/zip	zip archive
audio/mpeg	various
excel/*	Microsoft Excel
text/*	various
text/plain	various
text/richtext	various
text/rtf	Microsoft Word

VII Adding Support for Further MIME Types

7 Adding Support for Further MIME Types

Software AG supplies plugins that support a wide range of MIME types. However, you may need to process files that are not supported by these standard plugins. In that case, you can write your own plugin, using the information supplied in this section.

Developers who want to write their own plugins for the Tamino Non-XML Indexer should know how the software works, as described in the chapter **Architectural Overview**.

The main task consists of implementing the interfaces; these are fully described in the Javadoc documents that can be found in the directory *TaminoXNEDir/javadoc*. The necessary code to analyse the non-XML data and create XML data and metadata should be written as a Java program (JAR file); the class file or files must be added to the classpath, as described in the section **Configuring the Tamino Non-XML Indexer**.

If the plugin processes text data, it should be a subclass of `com.softwareag.tamino.nixe.plugin.AbstractTextIndexPlugin`.

If the plugin processes so-called binary data, it should be a subclass of `com.softwareag.tamino.nixe.plugin.AbstractBinaryIndexPlugin`.

In addition, you must make one or more additional entries in the mapping table that associates MIME types with plugins. This table is stored as the file *.../Tamino/Tamino Server Extensions/Java/com/softwareag/tamino/nixe/MimeTypePluginMapping.properties*. Simply add one line for each new MIME type that can be mapped. Use the existing data in the file as a basis when making your own modifications.

VIII Schema Reference Information

8 Schema Reference Information

The schema that is included with the Tamino Non-XML Indexer distribution kit has the structure described below. Properties generated by StarOffice and OpenOffice applications are mapped directly to the element contents and attributes of the schema.

```
<xdav_nonXML>
  <properties mimetype="xs:string" contentlength="xs:int">
    <content>
      xs:string <!-- contains the textual content of the non-XML object, possibly ↩
with XML tags -->
    </content>
    <meta>
      <office:document-meta version="xs:string">
        <office:meta>
          <meta:generator>           xs:string   </meta:generator>
          <dc:title>                 xs:string   </dc:title>
          <dc:description>           xs:string   </dc:description>
          <dc:subject>               xs:string   </dc:subject>
          <meta:initial-creator>     xs:string   </meta:initial-creator>
          <meta:creation-date>       xs:dateTime </meta:creation-date>
          <dc:creator>               xs:string   </dc:creator>
          <dc:date>                  xs:dateTime </dc:date>
          <meta:printed-by>          xs:string   </meta:printed-by>
          <meta:print-date>          xs:dateTime </meta:print-date>
          <meta:keywords>
            <meta:keyword>           xs:string   </meta:keyword>
          </meta:keywords>
          <dc:language>              xs:string   </dc:language>
          <meta:editing-cycles>      xs:string   </meta:editing-cycles>
          <meta:editing-duration>    xs:string   </meta:editing-duration>
          <meta:hyperlink-behaviour> xs:string   </meta:hyperlink-behaviour>
          <meta:auto-reload>         xs:string   </meta:auto-reload>
          <meta:template>            xs:string   </meta:template>
          <meta:user-defined name="xs:string"> <!-- zero or more occurrences -->
            xs:string
          </meta:user-defined>
```

```
          <meta:document-statistic attributes_are_tabulated_below />
        </office:meta>
      </office:document-meta>
    </meta>
  </properties>
</xdav_nonXML>
```

The element `meta:document-statistic` has the following attributes:

Attribute	Type
cell-count	xs:string
character-count	xs:string
draw-count	xs:string
image-count	xs:string
object-count	xs:string
ole-object-count	xs:string
page-count	xs:string
paragraph-count	xs:string
row-count	xs:string
table-count	xs:string
word-count	xs:string

Any number of `meta:user-defined` elements can be defined. Each element has an attribute "`name`". The content of the element defines the value. Example:

```
<meta:user-defined name="proofread-by">
  A.B. Campbell
</meta:user-defined>
<meta:user-defined name="proofread-date">
  31st April 2008
</meta:user-defined>
<meta:user-defined name="published-by">
  Westminster University Press
</meta:user-defined>
<meta:user-defined name="published-date">
  May 1st, 2008
</meta:user-defined>
```

Namespaces

Namespace	Description	URL
dc	The namespace "dc" refers to the Dublin Core metadata standard; for more information, please refer to the Dublin Core Metadata Initiative, **http://dublincore.com/**.	http://purl.org/dc/elements/1.1/
meta		http://openoffice.org/2000/meta
office		http://openoffice.org/2000/office

IX Mapped Properties

9 Mapped Properties

The schema, which is described in the previous chapter, applies directly to files produced by StarOffice and OpenOffice applications. Properties produced by other applications are mapped as described below.

This chapter lists all properties that are mapped to elements defined in the Tamino Non-XML Indexer schema (see the previous chapter entitled **Schema Reference Information**).

Microsoft Office

Microsoft Office property	Tamino Non-XML Indexer example
title	`<dc:title>My Title</dc:title>`
subject	`<dc:subject>My Subject</dc:subject>`

keywords	`<meta:keywords>`
	`  <meta:keyword>keyword1 keyword2... </meta:keyword>`
	`</meta:keywords>`

comments	`<dc:description>my comments</dc:description>`
author	`<meta:initial-creator>L.M. Name</meta:initial-creator>`
lastrevised	`<meta:user-defined ↩` `name="lastrevised">lastrevised</meta:user-defined>`
lastsavedatetime	`<dc:date>2003-05-05T16:00Z</dc:date>`
lastprinted	`<meta:print-date>2003-05-05T17:00Z</meta:print-date>`
edittime	`<meta:user-defined name="edittime">edittime</meta:user-defined>`
createdatetime	`<meta:creation-date>2003-05-04T08:00Z</meta:creation-date>`
application	`<meta:generator>notepad</meta:generator>`
template	`<meta:template>my template</meta:template>`
revision	`<meta:editing-cycles>2</meta:editing-cycles>`

Microsoft Office property	Tamino Non-XML Indexer example
pagecount	`<meta:document-statistic pagecount="320"/>`
lastauthor	`<dc:creator>P.Q. Name</dc:creator>`
manager	`<meta:user-defined name="manager">S.T. Name</meta:user-defined>`
category	`<meta:user-defined name="category">my category</meta:user-defined>`
company	`<meta:user-defined name="company">EZ ↩ Corporation</meta:user-defined>`

RTF (Rich Text Format)

RTF property	Tamino Non-XML Indexer example
created	`<meta:creation-date>2003-05-04T08:00Z</meta:creation-date>`
title	`<dc:title>My Title</dc:title>`
author	`<meta:initial-creator>L.M. Name</meta:initial-creator>`

PDF (Portable Document Format)

PDF property	Tamino Non-XML Indexer example
Date	`<dc:date>2003-05-05T16:00Z</dc:date>`
Title	`<dc:title>My Title</dc:title>`
Author	`<meta:initial-creator>L.M. Name</meta:initial-creator>`
Creator	`<meta:generator>notepad</meta:generator>`

Producer	`<meta:user-defined name="Producer">`
	`  G.H. Name`
	`</meta:user-defined>`

Note: The Tamino Non-XML Indexer supports the extraction of content and metadata from PDF files. However, in some circumstances, for example if the PDF file contains encrypted objects, no content information can be extracted.

MP3

MP3 property	Tamino Non-XML Indexer example
genre	`<meta:user-defined name="genre">classical</meta:user-defined>`
year	`<meta:user-defined name="year">1993</meta:user-defined>`
comment	`<dc:description>my comments</dc:description>`

album	`<meta:user-defined name="album">`
	`  Best of Bach`
	`</meta:user-defined>`

artist	`<meta:user-defined name="artist">`
	`  Glen Gould`
	`</meta:user-defined>`

title	`<dc:title>Partita No. 4</dc:title>`

Index