Xindice 1.0 Users Guide

$Revision: 1.1 $

by Kimbro Staken

1. Introduction to Xindice

1.1. What Is Xindice

The Xindice Core server is a database server designed from the ground up to store XML data. The Xindice server is what is termed by the XML:DB Initiative as a Native XML Database. You could also refer to it as a seamless XML database which might be an easier to understand description.

What this means is that to the largest extent possible you work with XML tools and technologies when working with the data in the server. All data that goes into and out of the server is XML. The query language used is XPath and the programming APIs support DOM and SAX. All things that should be familiar to a developer used to using XML in their applications. When working with XML data and Xindice there is no mapping between different data models. You simply design your data as XML and store it as XML.

What this gives you can be summed up in one word, flexibility. XML provides an extremely flexible mechanism for modeling application data and in many cases will allow you to model constructs that are difficult or impossible to model in more traditional systems. This model is a semi-structured model and for some applications it is an essential component. By using a native XML database such as Xindice to store this data you can focus on building your applications and not worry about how the complex XML construct maps to the underlying data store or trying force a flexible data model into a rigid set of schema constraints.

Ultimately though, Xindice is a tool. It will be right for some jobs and completely wrong for others. The job it is best at is simply storing XML data. In fact, that's all it does. If you have lots of XML data then Xindice may just be the right tool. However, if your data isn't XML or if you have the need for precise control over the structure of the data you might be better off using another database solution.

1.2. Current Status

Native XML database technology is a very new area and Xindice is very much a project still in development. The server currently supports storing well formed XML documents. This means it does not have any schema that constrains what can be placed into a document collection. This makes Xindice a semi-structured database and provides tremendous flexiblity in how you store your data, but, also means you give up some common database functionality such as data types. In its current state Xindice is already a powerful tool for managing XML data. However, there is still much that needs to be done. Feedback and contributions are actively encouraged.

This document attempts to describe those features that are working and can be used today. You should review the README file that is part of the Xindice distribution for the most current status on the project.

NOTE: Both the Xindice server and this document are works in progress. Any comments are welcome and encouraged.

1.3. Feature Summary

Document Collections: Documents are stored in collections that can be queried as a whole. You can create collections that contain just documents of the same type or you can create a collection to store all your documents together. The database doesn't care.

XPath Query Engine: To query the Document Collections you use XPath as defined by the W3C. This provides a reasonably flexible mechanism for querying documents by navigating and restricting the result tree that is returned.

XML Indexing: In order to improve the performance of queries over large numbers of documents you can define indexes on element and attribute values. This can dramatically speed up query response time.

XML:DB XUpdate Implementation: When you store XML in the database you may want to be able to change that data without retrieving the entire document. XUpdate is the mechanism to use when you want to do server side updates of the data. It is an XML based language for specifying XML modifications and allows those modifications to be applied to entire document collections as well as single documents.

Java XML:DB API Implementation: For Java programmers Xindice provides an implementation of the XML:DB API. This API is intended to bring portability to XML database applications just as JDBC has done for relational databases. Most applications developed for Xindice will use the XML:DB API.

XMLObjects: XMLObject provide a server extension mechanism for adding extra functionality to the server. They can be used to execute complex operations within the

database engine to cut down on network bandwidth or to add functionality that doesn't currently exist in the server.

Command Line Management Tools: To aid the administrator Xindice provides a full suite of command line driven management tools. Just about everything you can do through the XML:DB API can also be done from the command line.

CORBA Network API: For developers who are interested in working in languages other then Java, Xindice provides a CORBA API that can be used to build applications. All functionality available through the XML:DB API is also available through the CORBA API. In fact the XML:DB API is built on top of the CORBA API. Most Java developers though, will never even need to know that the CORBA API exists.

Modular Architecture: The Xindice server is constructed in a very modular manner. This makes it easy to add and remove components to tailor the server to a particular environment or to embed it into another application.

1.4. Database Structure

The Xindice server is designed to store collections of XML documents. Collections can be arranged in a hierarchy similar to that of a typical UNIX or Windows file system.

In Xindice the data store is rooted in a database instance that can also be used as a document collection. This database instance can then contain any number of child collections. In a default install of Xindice the database instance is called 'db' and all collection paths will begin with /db. It is possible to rename the database instance if desired though it is not necessary to do so.

Collections are referenced in a similar manner to how you would work with a hierarchical file system.

1.4.1. Collection Path Example

If you had a collection created under 'db' called my-collection and a collection under that called my-child-collection the path used when accessing the my-child-collection collection would be

```
/db/my-collection/my-child-collection
```

Within collections there are several types of objects that can be stored. You can store XML documents, XMLObjects and other collections. Each of these objects can also be referenced via a path.

1.4.2. Collection Path Referencing a Document

Extending the previous example by adding a document to my-child-collection named my-document it to could be referenced via a path.

```
/db/my-collection/my-child-collection/my-document
```

There is one catch to this however. Since you can have more then one object in a collection with the same name there is an order of precedence that is applied when evaluating a path. The order of precendence is collection followed by XMLObject and then document. What this means is that if you have a document and a collection with the same name you will not be able to retrieve the document.

You can also access collections on remote machines by specifying the host and port of the server.

1.4.3. Collection Path Referencing a remote Document

If the previous example was on a remote machine the path would look something like this.

```
myhost.domain.com:4080/db/my-collection/my-child-collection/my-document
```

This can also take the form of a Xindice URI.

```
xindice://myhost.domain.com:4080/db/my-collection/my-child-collection/my-document
```

1.5. Introducing the Command Line Tools

The Xindice server includes a command line program named `xindice` that allows you to manage the data stored in the server. A complete list of available commands and more detail about each command can be found in the Command Line Tools Reference Guide.

The `xindice` tool is located in the Xindice-Core/bin directory and it is probably a good idea to add this directory to your PATH environment variable. All examples in this manual will assume that the Xindice-Core/bin directory is on the operating system path.

2. Managing Documents

2.1. Introduction

In many ways the Xindice database can be viewed as a simple file store. This is of course a

highly simplified view of things but is a useful place to get started in learning the functionality of the server.

The Xindice server provides facilities to store, retrieve and delete well formed XML documents.

2.2. Adding Documents

2.2.1. Adding a Document With a Given Key

The document fx102.xml will be added to the collection /db/data/products and will be stored under the key fx102.

```
xindice add_document -c /db/data/products -f fx102.xml -n fx102
```

2.2.2. Adding a Document Without a Key

The document fx102.xml will be added to the collection /db/data/products. No key is provided so one will be generated automatically by the server. The generated key will look similar to this 0625df6b0001a5d4000bc49d0060b6f5

```
xindice add_document -c /db/data/products -f fx102.xml
```

2.3. Retrieving Documents

Documents can be retrieved from the database using the ID that they were inserted under.

2.3.1. Retrieving a Document Using an ID

The document identified by the key fx102 will be retrieved from the /db/data/products collection and stored in the file fx102.xml

```
xindice retrieve_document -c /db/data/products -n fx102 -f fx102.xml
```

2.4. Deleting Documents

2.4.1. Deleting a document using an ID

The document identified by the key fx102 will be removed from the collection

/db/data/products.

```
xindice delete_document -c /db/data/products -n fx102
```

3. Querying the Database

Xindice currently supports XPath as a query language. In many applications XPath is only applied at the document level but in Xindice XPath queries can be executed at either the document level or the collection level. This means that a query can be run against multiple documents and the result set will contain all matching nodes from all documents in the collection. The Xindice server also supports the creation of indexes on XML documents to speed up commonly used XPath queries. Please refer to the Administrators Guide for more detail about configuring indexes.

You can execute XPath queries against the database using the command line tools and the result of the query will be displayed.

3.1. Executing an XPath query against a collection of XML documents

Here we assume we have a collection /db/data/products that contains documents that are similar to the following.

```
<?xml version="1.0"?>
<product product_id="120320">
   <description>Glazed Ham</description>
</product>
```

The XPath /product[@product_id="120320"] will be executed against the collection /db/data/products and all matching product entries will be returned.

```
xindice xpath_query -c /db/data/products -q /product[@product_id="120320"]
```

The result of the query is an XPath node-set that contains one node for each result. In this particular example there is only one result and the node that matches is the root element so you get back basically the whole document.

To make it easy to link the result node back to the originating document, Xindice adds a few attributes to the result. These attributes are added in the NodeSource namespace that has the URI http://xml.apache.org/xindice/NodeSource. The `col` attribute specifies the collection where the document can be found and the `key` attribute provides the key of the original document. Using this information it is possible to retrieve the original document that this node was selected from for further processing.

```
<product product_id="120320" xmlns:src="http://xml.apache.org/xindice/NodeSource"
```

```
      src:col="/db/data/products" src:key="120320">
   <description>Glazed Ham</description>
</product>
```

If more then one result is found the results look something like this. This could be the result of the query /product

```
<product product_id="120320" xmlns:src="http://xml.apache.org/xindice/NodeSource"
      src:col="/db/data/products" src:key="120320">
   <description>Glazed Ham</description>
</product>
<product product_id="120321" xmlns:src="http://xml.apache.org/xindice/NodeSource"
      src:col="/db/data/products" src:key="120321">
   <description>Boiled Ham</description>
</product>
<product product_id="120322" xmlns:src="http://xml.apache.org/xindice/NodeSource"
      src:col="/db/data/products" src:key="120322">
   <description>Honey Ham</description>
</product>
```

While it is certainly useful to be able to query from the command line it is probably more useful to be able to use the results of a query in an application. For more information on building applications for Xindice, please refer to the Developers Guide.