

# Apache Xindice 1.0 History of Changes

\$Revision: 1.2 \$

## 1. Releases 0.1 to 1.0 History of Changes

### Apache Xindice Version 1.0

=====

This is the first production release of Xindice. Changes from the last release candidate are minimal.

- Fixed a path traversal security problem in the HTTP server.
- Fixed the Addressbook example to not send data to the client after the connection had already been committed.
- SAXGenerator now properly generates prefixMapping events.

Known issues in version 1.0:

- UTF-8 Encoding is not entirely clean. Most latin derived languages should be OK, but English is the most robust. Xindice 1.1 will resolve any issues in this area.
- XPath queries that return a single atomic value (i.e. the value of an attribute) rather than a node will return no result. You must retrieve the containing element to retrieve the content of an attribute.
- When using XUpdate with JDK 1.4 you must use the standards override mechanism to replace the version of Xalan included in the JDK with the version included in Xindice.  
See: <http://java.sun.com/j2se/1.4/docs/guide/standards/index.html> for more information.
- On Windows, command line queries can have problems with the quote handling of the windows shell. In general you should put double quotes around the entire query string and use single quotes in your XPath.
- This initial release of Xindice does not have any built in security. If you run it on a public server you should insure that remote access to port 4080 is restricted at the network level. Security will be added in a future release.

### Apache Xindice Version 1.0rc2

=====

The focus of this release is on stabilization of the server.

- Fixed the Index corrupted error that some people were seeing with 1.0rc1. If you saw this error it is recommended that you rebuild your database files.

## Apache Xindice 1.0 History of Changes

- Changed the way Xindice locates its files to make it easier to embed the server into another process. Files are now located relative to the xindice.home system property instead of the working directory of the process.
- Changed the kernel to enable running it embedded without exiting the VM on startup error and exit.
- Minor encoding fixes in the command line tools. More serious attention will be paid to encoding issues in the 1.1 release of Xindice. As it is some languages such as Russian and Chinese can not be successfully stored in the server. This will be fixed in a Xindice 1.1 release.

### Apache Xindice Version 1.0rc1

=====

dbXML is now an Apache project, and has been renamed to Xindice (Zeen-dee-chay). Parts of the dbXML 1.5 tree were merged into the dbXML 1.0 tree in the process of this name change and migration, so we thought it best to release at least one release candidate as an Apache project. There are also many changes as a result of the branch merging.

- Name changes. There have been a lot of changes in package, class, documentation, and identifier naming throughout the project as a result of the migration to the Apache project. The most important are summarized here.
  - XML:DB URI changes. All XML:DB API uri should now be of the form xmldb:xindice: instead of xmldb:dbxml:
  - Source package changes. If you have any code that imported any org.dbxml.\* source code it will need to be changed to import the proper packages from org.apache.xindice.\*.
  - XML Namespace changes. XML namespaces that were defined by dbXML have been renamed. The "http://www.dbxml.org/" portion of those namespaces has been changed to "http://xml.apache.org/xindice/".
- The Collection configuration system now uses the Database's system collection instead of the system.xml file. The system.xml file is now read-only, and is used for configuring the server framework. Collection management is read/write and uses the Xindice native file system to maintain configuration.
- As installed the server no longer has any default collections that can store documents. You must create a collection manually before attempting to store any documents in the server.
- Complete JAXP bootstrapping. Xindice will bootstrap with whichever JAXP-capable XML parser the Java VM will resolve. You can override the JAXP SAXParserFactory using vm.cfg. It is not recommended that you override the JAXP DocumentBuilderFactory because Xindice implements an optimized DOM that utilizes the Xindice compression system.

## Apache Xindice 1.0 History of Changes

- Lazy writes have been added to the Paged system, which is the foundation for standard Filers and Indexers in Xindice. Long operations (like index creation) will now delay writes until the write buffer is filled or until the operation is completed. This can yield a 10% to 30% performance increase on index creation.
- The --pagesize and --maxkeysize switches now work on Collection creation in addition to Index creation.

### Version 1.0b4 (Final Beta... No Really)

=====

After releasing beta 3, we found out that there were some stability issues with the latest developer releases of Xalan, whose XPath engine we use for our query resolver. Some users were experiencing query failures with certain data sets. Because of this, we've had to roll back to a previous version of Xalan (2.0.1).

### Version 1.0b3 (Final Beta)

=====

Beta 3 is the final beta for dbXML before we release our 1.0 FCS version. This version provides improved concurrency, as well as several bug fixes.

### Version 1.0b2 (Beta 2!)

=====

Improved stability and scalability of the server.

- ORB Change. In the past JacORB was used as the dbXML CORBA ORB, with this release JacORB has been replaced with OpenORB. It was found JacORB utilized too much memory while running as part of the server which severely limited the capacity of the system.
- The XML:DB API has once again been brought in to conformance with the latest draft.
- Several DOM Level 3 Core methods have been added, and the version of Xerces shipped with dbXML is now the most recent version in the Xerces 1 distribution.
- Several bugs within the XUpdate system have been fixed.

### Version 1.0b1 (Beta!)

=====

We have reached Beta status. The server is fully functional, and the number of bugs should be minimal at this point.

- Namespace support. The query and indexing systems now properly support namespaced elements and attributes (regardless of prefix

consistency).

- The most recent draft of the XML:DB API is now supported. This includes namespace support for XPath queries, and a few minor changes to the API.
- A Testing framework has been added under java/tests. It is based on junit and can be used to perform regression testing against the server.
- GZip compression was removed from the filers. It was slow. Also, because it was both buggy and out of our control, we had to get rid of it.
- Lots of little bugs fixed here and there.

#### Version 0.9.1 (The Broken ORB)

=====

Some minor updates, nothing to be alarmed about. Move along.

- The XUpdateQueryService is now available via the XML:DB Collection class.
- A lot of the problems that were being reported regarding ORB versioning and VM configuration have been resolved.
- Our DOM was broken in respect to DocumentFragments. Also, a bug in reporting node modification status up the tree has been fixed. This was causing XUpdate queries to break in some cases.
- The Exception system has been further refined.

#### Version 0.9 (Feature Complete)

=====

Several major changes have happened to the dbXML code base between versions 0.6 and 0.9. The most important of which is that we are now feature complete.

- We are now feature complete. All of the features that will be in the 1.0 version of dbXML are now available. All we have to do now is continue to stabilize the server and fix bugs as they pop up. You can consider the status of the project to be Alpha quality now.
- dbXML is now based on an Apache style license. We decided that the LGPL was too restrictive regarding what you could do with the source code. Beyond that, we're using several BSD and Apache licensed libraries, and it seemed unfair that we could build from their code, but they couldn't build from our's.
- dbXML now includes support for the XML:DB XUpdate specification. We've integrated The Infozone Group's Lexus library into dbXML

## Apache Xindice 1.0 History of Changes

in order to provide support for XUpdate update logic.

- Wire Compression is now supported by the CORBA APIs. The style of compression that is used by our DOM and SAX classes for Document storage is now being exposed via CORBA. This allows Documents and query results to be retrieved without requiring textual serialization on the server or parsing on the client. This capability is transparently supported by the XML:DB API.
- NodeIndexer has split into NameIndexer and ValueIndexer. The ValueIndexer is used as NodeIndexer was, to store values for predicate comparisons. NameIndexer is used to store element references for standalone name components in location paths. Use a type of 'name' when defining an Index to create a NameIndexer.
- Better Exception categorization. Exception fault codes have been further defined, categorized, and broken out by severity. The FaultCodes class now includes several utility methods for generating APIException instances and examining the fault codes that are stored in various types of Exception classes.
- Application has been renamed to Database. Also, references to Application in various methods need to be changed to Database. Ex: getApplication() is now getDatabase()
- An Address Book example is included, built on Tomcat. You can find more information in java/examples/Addressbook/README
- The CORBA ORB used by the server is now easily pluggable. So far, JacORB and OpenORB are known to work.
- SAX support has been added to the XML:DB API implementation.
- The HTTP server port has been changed to 4080 to avoid conflicts on the commonly used 8080 port. This is mainly because Tomcat uses that port. Also, the Gopher port has changed to 4070.
- More and more documentation.

### Version 0.6 (Much Closer)

=====

In the past couple of weeks, we've made quite a bit of progress in building out the server, and contributing to its overall stability. There's a lot left to do, but it's getting very close to being usable.

- Lots of bug fixes in this version, but many more to come.
- The Developer's Guide has been fleshed out quite a bit, and the Command-line Tools reference has been updated and converted to DocBook format.

- dbXML fully supports the XML:DB API as it is currently published by the XML:DB Initiative. XML:DB API documentation is now included in the distribution.
- Types are now supported by the NodeIndexer to ensure proper sorting. The available types can roughly be mapped to the Java native types (string, short, int, etc...)
- The XPathQueryResolver now supports partial evaluation of some functions and index-based evaluation of the starts-with function.
- The XPathQueryResolver also supports the highly experimental, very cool, and potentially catastrophic autoindex feature. By default, it's turned off, so there's nothing to worry about.
- IndexManager now performs background indexing instead of synchronous. Issuing a create index command will now immediately return as successful even though the index itself hasn't yet been built.
- Query results now include a set of namespaced attributes that identify the collection and document that a particular node was retrieved from.
- The command line tools now require an instance name when referencing a collection. The default instance name in a dbXML server is 'db'. So, for example, you might refer to a collection as '/db/root/addressbook'. Also, the short form of some of the action verbs have changed. See the Tools reference for more information.

#### Version 0.5 (Woah!)

=====

We've made some major changes to dbXML between version 0.4 and 0.5 that will affect the type of applications that can be developed solely with dbXML, so it's important to read this change log for more information.

- dbXML has been broken into three separate projects, with the development focus remaining on the dbXML Core database server. Two other projects: The Juggernaut Server Framework, and dbXML App Services are available as separate CVS trees and are being developed in parallel. The Juggernaut class files are available in a Jar file as part of the distribution. The following is a list of the features that have been removed from the dbXML Core, and where they are now:
  - Juggernaut - cvs co Juggernaut
    - Service Framework
    - HTTP Server
  - App Services - cvs co dbXML-AppServices
    - GetObject (HTTP Retrieval)
    - SOAP Support

## Apache Xindice 1.0 History of Changes

- Cocoon Support
  - Scripting Support
  - Schema Compiler
  - XMLObject Compiler
- We've renamed our packages from com.dbxml.\* to org.dbxml.\*
- The ENTIRE Filing, Indexing, and Query systems have been completely rearchitected and rewritten pretty much from scratch. As a result:
- QueryResolvers can be developed and plugged into the QueryEngine.
  - Full XPath syntax is now supported for Collection queries. This functionality is provided by the XPathQueryResolver.
  - The Indexing system participates in queries wherever possible.
  - You can safely add and remove Indexes to existing Collections.
- A new Filer named BTreeFiler is available in addition to HashFiler. BTreeFiler is much more space conservative and doesn't suffer from collision and overflow issues as the Collection begins to grow past its original bounds. Both Filers are useful, but which you choose depends on your needs. By default, dbXML core will use BTreeFilers.
- The Application class now extends Collection and can be thought of as a top-level root Collection. At some point in the future, Application will be renamed Database.
- There have been a few changes to the Collection class. You can no longer store binary data in a Collection, only Documents. The getDocumentSet method allows you to enumerate through the Documents in a particular Collection. Collection has been broken into two classes. CollectionManager contains all management functionality for nested Collections (create, drop, list) while Collection contains functionality for the Collection instance (getDocument, insertDocument, etc...)
- XMLObjects have been scaled back. There is now only one type of XMLObject. Application and Document XMLObjects have been removed. Because Application is now derived from Collection, a standard XMLObject can serve both roles. Document XMLObjects have been removed completely, requiring a developer to implement this functionality manually (it's about 1 line of code). The mapping looks like this:
- ```
ApplicationContext -> XMLObject
ApplicationXMLObject -> (gone)
CollectionContext -> XMLObject
CollectionXMLObject -> SimpleXMLObject
DocumentContext -> (gone)
DocumentXMLObject -> (gone)
```
- The dbXML Client API has been replaced by an XML:DB Core Level 1 implementation. The XML:DB API is still a work in progress, and

is likely to change, but this opens the doors to interoperable XML Database applications. For more information on the XML:DB API, visit <http://www.xmldb.org>

- The Command-Line Tools have been broken into two separate tools. dbxmladmin provides administrative commands, while dbxml provides user-level commands. The Command-Line Tools now utilize the XML:DB API instead of the Client API. Some new features in the Command-Line Tools include:
  - Server Shutdown - You can now safely shut down the server, instead of having to send it a KILL signal.
  - Import/Export - You can import/export multiple Documents and directory structures between Collections and the file system.
  - XMLObject invocation. You can execute XMLObject methods and retrieve their results.
- We're now using JacORB for our CORBA services. The JDK's ORB was very much lacking in a lot of areas.
- JAXP support for creating and parsing dbXML compressed DOM Documents is now available.
- And a whole bunch of other stuff.

#### Version 0.4 (Progress)

=====

We've made quite a bit of progress between version 0.3 and 0.4 in features and in general system stability and performance.

- The Indexing System and XPath querying are working. The indexing system now allows you to specify a XPath for narrowing individual indexes.
- The Compressed DOM is essentially complete.
- We've integrated Cocoon into dbXML to maximize transformation performance.
- XMLObjects can now be created at various contexts within the server. These are Application, Collection, and Document. The ability to associate business logic at various levels of the repository is a powerful application design/management capability.

As part of this:

- What used to be XMLObjects are now DocumentContext XMLObjects.
- What used to be Procedures are now CollectionContext XMLObjects.
- Nested Collections. You can now manage collections of documents in a nested fashion for logically laying out your data stores. Databases have been replaced by top-level Collections.
- The SystemCollection class will automatically compile a Schema



## Apache Xindice 1.0 History of Changes

using the XMLSchemaCompiler upon calling the setSchema() method.

- XMLSerializable objects are classes whose state can be serialized to and from XML documents. The serialization is not an automated process at the moment, but the ability to introspect an object graph and produce XML is planned for a future release. XMLSerializable objects can be stored/retrieved to/from the database with the Collection set/getObject methods.

As part of this:

- SymbolTables are now represented using XMLSerializable objects.
  - Schemas are now represented using XMLSerializable objects.
  - XMLSchemaCompiler now produces XMLSerializable objects.
- A Compressed DOM Symbol Table can be defined in the system configuration for hard-coding or using standardized symbol tables. SystemCollection uses a hard-coded symbol table to store compressed symbol tables.
  - A Gopher Service is now available, allowing Gopher-based directory and document browsing and querying of a dbXML repository. Gopher is useful for quickly browsing to documents being stored in the repository.

### Version 0.3 (Bye Bye C++)

=====

The C++ code is gone. dbXML is now 100% Java code. There have also been a few major additions to the system:

- More Documentation. Yippee!
- The Configuration framework is essentially fully functional.
- The Compressed DOM is functional but still in an experimental state. A compressed Collection can be created by setting the compressed attribute to 'true' in the collection element. There are still some missing implementations, especially where DTD types are concerned, but most of the document core should work.
- The foundation for dbXML autolinking is part of the dbXML Compressed DOM system. dbXML will automatically expand elements with links and respect document caching policies in expanding those links. See the User's Guide for more information.
- The Indexing system is getting much closer to completion. Basic XPath querying is also in an experimental state.
- A command line tool for managing the running server. This uses the CORBA APIs to manage the server.
- XML Schema Compiler - The XML Schema compiler takes a W3C XML Schema (xsd) resource, and generates a set of Java classes based on the element, attribute, and element-relationship definitions in the

Schema. The compiler still needs a lot of work in order to generate typed attributes (right now everything is a string), but it's a good start. In the future, this compiler will be an internal process, compiling all stored schemas for utilization by XMLObjects (so you don't have to use the DOM directly).

- SOAP Support - All XMLObject and Procedures are automatically exposed by the server as SOAP services (as well as their original native protocol). SOAP support is limited to the capabilities of Procedures and XMLObjects. Object structure serialization may be implemented in a future release.

#### Version 0.2 (Switch To Java)

=====

A major architectural shift occurred between 0.1 and 0.2. A design that had once consisted of about a 90%/10% C++ to Java ratio, has flip-flopped to a 95%/5% Java to C++ ratio. There are several reasons for this. First, in order to provide better integration with existing open source XML Server architectures, which are almost all Java-based, we decided that it would be best to avoid mixing the Java and non-Java worlds wherever possible. Second, we would be able to afford ourselves a major kick-start by utilizing some of the better parts of the Juggernaut architecture in our design. Third, doing XML in C++ is a headache. You spend more time worrying about memory management than you do in actually writing functioning code. In order to maintain a certain level of sanity for our staff, and contributors to the dbXML source code, we decided that Java would be the best choice for an implementation language.

#### Version 0.1 (In The Beginning)

=====

The Three Filing Systems are sort of finished. There are likely a lot of places to optimize them and there are absolutely some re-entrant code issues, but these will be ironed out as I actually start using the filers with the Parser and Query Engine. HashFiler is a disk-based hashed bucket filing system. FSFiler is a filer that loads data directly from the operating system's file system based on their file name. MemFiler is a memory-based filing system, mainly for temporary in-memory tables and query result sets.

A quick note about the HashFiler. dbXML's filing system was not written to be disk-space conservative, it was written to be incredibly efficient for handling large, variable-sized chunks of data. Where systems like gdbm and dbm try to be everything to everyone, HashFiler is really targeted for the dbXML project. HashFiler provides a simple block read caching mechanism with a default size of 50 blocks. All writes are performed immediately.

Blocks should generally be optimized to a multiple of the operating system's block size and the number of pages per block should be a power of 2 and the resulting size of a page should be large enough to store the PageHeader (~64 bytes), key (up to you), and at least a

## Apache Xindice 1.0 History of Changes

fair amount of record data.

HashFiler supports record compression if the size of a record spans past a single block and if compression will actually yield a compressed value (meaning if the compression actually lengthens the record, the compression is canceled). Compression can be toggled with the `setCompressed` method and tested with the `isCompressed` method. If a HashFiler has operated for a time with compression and compression is then turned off, existing compressed records are not decompressed until rewritten. Compression is performed using `zlib` with a compression method set to `Z_BEST_SPEED` which will perform well against variable length textual data (such as XML documents) but not most binary data.

### Acknowledgments

-----

This product includes software developed by the Infozone Group  
(<http://infozone-group.org>)

This product includes software developed by the XML:DB Initiative  
(<http://www.xmldb.org>)

This product includes software developed by the Exolab Project  
(<http://www.exolab.org>)