Oracle Berkeley DB XML

Programmer's Reference Guide

12c Release 1

Library Version 12.1.6.0

ORACLE

BERKELEY DB

Legal Notice

Published 7/10/2015

Table of Contents

Preface

Welcome to Berkeley DB XML (BDB XML). This document introduces BDB XML, version 6.0. It is intended to provide a rapid introduction to the BDB XML API set and related concepts. The goal of this document is to provide you with an efficient mechanism with which you can evaluate BDB XML against your project's technical requirements. As such, this document is intended for C++ developers and senior software architects who are looking for an in-process XML data management solution. No prior experience with Sleepycat technologies is expected or required.

Conventions Used in this Book

The following typographical conventions are used within in this manual:

Class names are represented in `monospaced font`, as are `method names`. For example: "The `XmlDatabase::openContainer()` method returns an `XmlContainer` class object."

Variable or non-literal text is presented in *italics*. For example: "Go to your *DBXML_HOME* directory."

Program examples are displayed in a monospaced font on a shaded background. For example:

```
#include "DbXml.hpp"

using namespace DbXml;
// exception handling omitted for clarity

int main(void)
{
    // Open an XmlManager.
    XmlManager myManager;
}
```

For More Information

Beyond this manual, you may also find the following sources of information useful when building a BDB XML application:

- Introduction to Berkeley DB XML
- Berkeley DB XML Getting Started with Transaction Processing for C++
- Berkeley DB XML C++ API Reference Guide

To download the latest Berkeley DB XML documentation along with white papers and other collateral, visit http://www.oracle.com/technetwork/indexes/documentation/index.html.

For the latest version of the Oracle Berkeley DB XML downloads, visit http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/downloads/index.html.

Contact Us

You can post your comments and questions at the Oracle Technology (OTN) forum for Oracle Berkeley DB XML at: https://community.oracle.com/community/database/high_availability/berkeley_db_family/berkeley_db_xml.

For sales or support information, email to: berkeleydb-info_us@oracle.com You can subscribe to a low-volume email announcement list for the Berkeley DB product family by sending email to: bdb-join@oss.oracle.com

Chapter 1. Introduction to Berkeley DB XML

Berkeley DB XML (BDB XML) is a programmatic toolkit specifically designed to store and manage XML data in its native format. BDB XML is built on top of the existing Berkeley DB database product, which provides fast, reliable, scalable, and mission-critical database support. Application developers can choose the version of Berkeley DB that is most suitable for a particular application: Berkeley DB Data Store, Berkeley DB Concurrent Data Store, Berkeley DB Transactional Data Store, or Berkeley DB High Availability.

BDB XML provides document query support through XQuery 3.0 and, by extension, XPath 2.0. XQuery is a W3C Recommendation. BDB XML also supports the XQuery Update for modification of document content. In addition, BDB XML is tested against the XQuery Test Suite, version 1.0, and results have been published to the W3C.

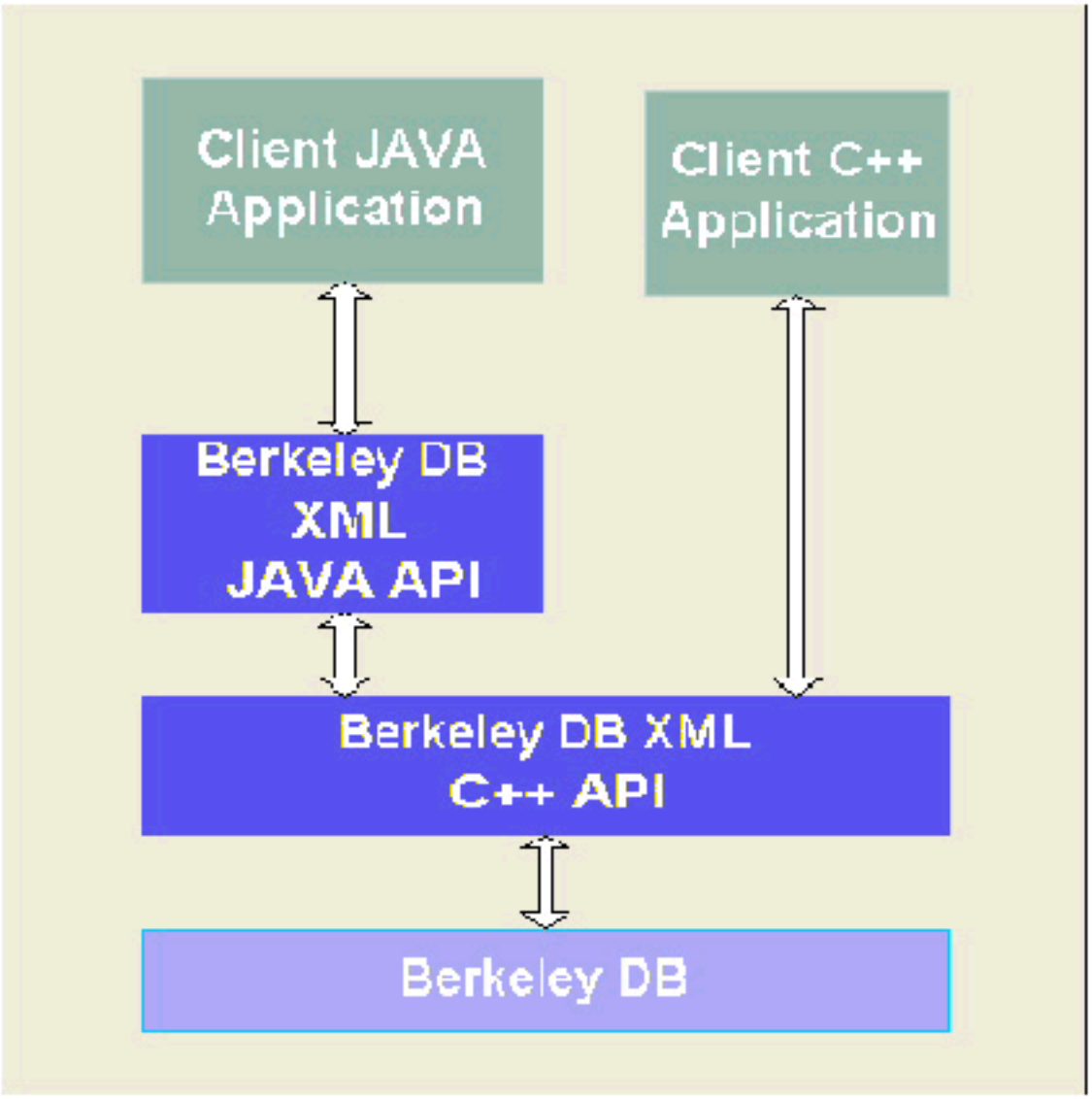
This document provides a very high level introduction to BDB XML. Users of BDB XML are assumed to have existing knowledge of XML, XQuery, XPath, either C++ or Java, and Berkeley DB.

This document also provides instructions on how to build the library, and instructions on how to compile and link the library with your application.

For a brief tour of Berkeley DB XML, see Introduction to Berkeley DB XML. For a complete introduction to BDB XML, see either the C++ or Java version of the Berkeley DB XML Getting Started Guide. For a complete description of the BDB XML API, see either the c++ api reference; or the Javadoc.

Architecture

Berkeley DB XML is implemented as C++ library on top of Berkeley DB. BDB XML is distributed as a shared library that is embedded into the client application. The BDB XML library exposes API's that enable C++ and Java applications to interact with the XML data containers. Figure 1 illustrates the Berkeley DB XML system architecture.



BDB XML uses Berkeley DB for data storage and transaction management. Client applications can also store data directly to a Berkeley DB database. Although BDB XML hides much of the internal use of Berkeley DB, some understanding of the underlying Berkeley DB API is required, as some BDB XML API methods accept Berkeley DB object handles as parameters. In particular, transactional applications need to fully understand the Berkeley DB database management interfaces for operations such as backup and restore, archiving, database recovery, etc.

The BDB XML library comprises several main components: document storage, XML indexing and index management, query optimization, and query execution.

Document Storage

Within Berkeley DB XML, documents are stored in containers. Containers are named and are files that include a number of Berkeley DB databases for information such as documents, indices and index statistics, data dictionary, and other system metadata. A container is the scope for indices, document names, container type, and other container-specific information. A client application can operate on multiple containers concurrently, and controls the placement of documents within containers. The client application can also store data to Berkeley DB databases. A client application can perform the following actions against a container:

- Create or remove a container.
- Add or drop an index in a container.
- Open a container for use within the application.
- Insert or delete a document in a container.
- Retrieve a document from a container.
- Update an existing document entirely or in part.
- Set, modify, or remove document metadata.
- Query a container using an XQuery or XPath expression.
- Close a container.
- Rename documents and containers.
- Dump a container to a text file.
- Load a container from a text file that was generated by a container dump.
- Verify that a container is internally consistent.

For a complete description and examples of how to use the BDB XML API to perform these tasks, see either the C++ or Java version of the Berkeley DB XML Getting Started Guide.

Chapter 2. Building Berkeley DB XML for UNIX/POSIX systems

The Berkeley DB XML distribution comprises several libraries: a base C++ library, three third-party libraries, and optional libraries for additional language interfaces such as Java, Perl, Python, PHP, and Tcl. Instructions for building the base libraries as well as Java and Tcl interfaces are included here. Instructions for the other language bindings appear in their respective directories:

```
dbxml-6.0.xx/dbxml/src/{perl,php,python}
```

All bindings require the C++ library and third-party libraries. Building for Linux, Mac OS X, and Cygwin (Windows) is the same as building for a conventional UNIX platform.

Using the buildall.sh Script

The Berkeley DB XML distribution uses the Free Software Foundation's autoconf and libtool tools to build on UNIX platforms. In general, the standard configuration and installation options for these tools apply to the Berkeley DB XML distribution. For ease of use Berkeley DB XML uses a shell script called `buildall.sh` to drive the build process. It wraps the `configure` and `make` steps associated with `autoconf`.

Preparing to build

Berkeley DB XML makes use of several third-party libraries. Berkeley DB XML provides a shell program that makes it possible to build all of the libraries in one command. The Berkeley DB XML distribution includes the complete source for compatible versions of the third-party libraries. No additional downloads are required. Use of other versions may not work, and may not be supported.

Each of these packages is freely available and distributed under an Open Source license, although Berkeley DB XML depends on specific versions of each. To build Berkeley DB XML, you will need the following third-party libraries:

- Berkeley DB

  Berkeley DB is a general purpose database toolkit. This release of Berkeley DB XML includes Berkeley DB 6.1.x release, and requires the Berkeley DB 4.3.28 release or later.

- Xerces

  Xerces is the Apache implementation of XML parsing technology and DOM. The build requires a source release of Xerces, not just a binary release. This release of Berkeley DB XML bundles Xerces-C 3.1.2, and requires the Xerces 3.0.1 release or newer.

- XQilla

  XQilla is an open source partial implementation of XQuery 3.0 and XPath 2.0. Berkeley DB XML bundles a specific public release of XQilla which must be used. Later versions of XQilla may work, but are not implicitly supported.

It is necessary to use GNU make (gmake) to build Berkeley DB XML and its third-party libraries. If you do not already have GNU make on your system, you will need to install it. When using gcc/g++, Berkeley DB XML requires at least a 3.x release of gcc.

Building Berkeley DB XML and third-party libraries

These instructions assume that you are building in the Berkeley DB XML distribution. The simplest possible build command is the following:

```
cd dbxml-6.0.xx
sh buildall.sh
```

This will configure, build, and install all of the libraries using default settings, which installs them into the following directory:

```
dbxml-6.0.xx/install
```

The default build supports the C++ interface only. The default build also compiles Berkeley DB XML examples, and places them in the following directory:

```
dbxml-6.0.xx/dbxml/build_unix
```

The `buildall.sh` script has a number of configuration options. You can see them with this command:

```
sh buildall.sh --help
```

- `--enable-debug` to build debug libraries
- `--prefix=path` to change the default common installation path (from `./install`)
- `--enable-java` to build the Java API. This also requires that there is a working `javac` in your `PATH`. The resulting .jar files are created in the `install/lib` directory. Java examples are also built, and put in:

  ```
  dbxml-6.0.xx/dbxml/build_unix/dbxmlexamples.jar
  ```

- `--enable-perl` to build the Perl interface
- `--disable-compression` to disable default compression
- `--clean` to clean (make clean) the entire build
- `--distclean` to clean (make distclean) the entire build. This option clears the configuration state as well as the compilation state.

There is no option for per-library clean or rebuild in buildall.sh. See Building Individual Berkeley DB XML Libraries for UNIX (page 8) for information on how to work with each library build.

Configuring Berkeley DB XML and third-party libraries

The `buildall.sh` script has a number of configuration flags to handle platform-specific issues, change target and destination directories, as well as product configuration options.

The flags that start with "`--with-`" apply to each of the libraries, depending on the name used. In these examples, "libname" is used to represent one of "dbxml," "xerces," "xqilla," or "berkeleydb." For example, a valid flag may be "`--with-berkeleydb-prefix=path`" or "`--with-xerces-prefix=path`." The following is a partial list of options:

- `--help`

  Use this flag to return a help message for buildall.sh.

- --enable-debug

  Use this flag to build with `-g` as a compiler flag and with DEBUG #defined during compilation. This will create libraries and utilities with debugging symbols. This argument should not be specified when configuring to build production binaries.

- `--enable-java`

  Use this flag to build with Java support. This builds Java support for both Berkeley DB and Berkeley DB XML. It is necessary to ensure that there is a working javac in your PATH. The resulting .jar files are created in the `install/lib` directory. Java examples are also built, and put in:

  ```
  dbxml-6.0.xx/dbxml/build_unix/dbxmlexamples.jar
  ```

- `--enable-perl`

  Use this flag to build with Perl support. This builds Perl support for both Berkeley DB and Berkeley DB XML. This option requires a working perl program in your PATH.

- `--with-tcl=path`

  Use this flag to build with Tcl support. `path` is the directory in which the Tcl tclConfig.sh file may be found. See `Loading Berkeley DB with Tcl` in the *Berkeley DB Programmer's Reference Guide* for information on sites from which you can download Tcl and which Tcl versions are compatible with Berkeley DB XML.

- `--disable-compression`

  Disables default compression. Compression is turned on by default. Default compression requires the library ZLIB. --with-zlib can by used to specify the path to the ZLIB installation.

- `--enable-static-zlib`

  Causes the ZLIB library to be linked statically instead of dynamically.

- `--with-zlib=path`

  Default compression requires the ZLIB library. If ZLIB is not installed in the directory /usr/local then use this argument to specify the directory in which it is installed. If no ZLIB installation can be found then default compression is disabled automatically.

- `--enable-test`

Enables test suite support. This option requres --with-tcl and --enable-debug, and if using --enable-java it also requires --with-junit.

- `--with-junit`

  Path to a JUnit (4) .jar file. This option is required if both --enable-test and -enable-java are specified.

- `-p platform`

  To specify a platform type to the Xerces runConfigure script, use -p platform. For most platforms, configure guesses the type correctly. Use this option only if there are configuration errors without it.

- `-c C compiler name`

  To build with C compiler other than gcc, specify the name of the compiler, for example, -c cc.

- `-x C++ compiler name`

  To build with C++ compiler other than g++, specify the name of the compiler, for example, -x cxx.

- `-m make_command`

  To build with make program other than `make`, specify the name of the program. For example, -m gmake.

- `--no-configure`

  To build without re-running the configure step for each library, enter --no-configure as an argument to buildall.sh.

- `--clean`

  To clean compilation and linking object from all libraries, use --clean.

- `--distclean`

  To clean configuration state from all libraries, use --distclean. After this is done, configuration must be run again in order to build.

- `--build-one=library`

  To build, or clean only one of the libraries, use --build-one=library, where library is one of berkeleydb, xerces, xqilla or dbxml.

- `--prefix=path`

  To change the common installation directory prefix from the default (./install), specify an alternate path.

- `--with-libname=path`

  To use library sources from a location other than the default, specify a path to the source. For example, if Xerces source were located in /home/xerces-c-src_3_1_1, specify, --with-xerces="/home/xerces-c-src_3_1_1"

- `--with-libname-prefix=path`

  To change the installation directory for only a single library, specify the path using --with-libname-prefix=path. For example, if Berkeley DB is to be installed in `/usr/local/BerkeleyDB.6.1`, specify:

  ```
  --with-berkeleydb-prefix="/usr/local/BerkeleyDB.6.0"
  ```

- `--with-libname-conf=configure flags`

  All of the library configurations use the configure program. Use --with-libname-conf=configure flags to pass library-specific configuration flags.

Advanced Building and Troubleshooting

If `buildall.sh` does not provide enough control for your environment, or if you have trouble with any of these commands, please see the next section, which goes into more detail on build options. At the end of that section, there is information on what to do if build problems persist.

Building Individual Berkeley DB XML Libraries for UNIX

The `buildall.sh` script is a wrapper around separate build procedures for each third-party library. If there are problems using the script, or if it is necessary to build libraries individually, it is possible to build manually. Building manually gives you finer control over configuration, build and installation. If you are able to build successfully using buildall.sh, the information in this section is not necessary.

Building Individual Libraries

These instructions assume that the third-party libraries are in the directory, dbxml-6.0.xx. They apply regardless of actual location.

Berkeley DB

Berkeley DB XML requires Berkeley DB 4.3.28 or newer. This release includes Berkeley DB 6.1.x. Adjust the path below to match the version bundled with the distribution.

This will install Berkeley DB in `/usr/local/BerkeleyDB.6.1`. If you want to install in a different directory, add `--prefix=/path/to/install` to the configure line.

If you want to use the Berkeley DB XML Java API, you must also build Berkeley DB with Java support by adding `--enable-java` to the configure line.

Xerces

Xerces is the Apache implementation of XML parsing technology and DOM. The build requires a source release of Xerces, not just a binary release. This release of Berkeley DB XML bundles Xerces-C 3.1.2, and requires the Xerces 3.0.1 release or newer.

Example Bourne shell build commands for Xerces are:

```
cd dbxml-6.0.xx/xerces-c-src
export XERCESCROOT=`pwd`
cd src/xercesc
./configure --disable-pretty-make
make
make install
```

The option --disable-pretty-make is added because that eliminates the requirement for GNU make. It results in more verbose output. If it is not used GNU make will be required.

This will install Xerces under `/usr/local`. If you want to install in a different directory, add `-P/path/to/install` to the configure line.

XQilla

XQilla is a partial implementation of XQuery 3.0 and XPath 2.0. This release of Berkeley DB XML bundles the XQilla version that is required. Other versions may work, but are not implicitly supported. Generally releases with the same major and minor number will be compatible.

An example Bourne shell build command for XQilla is:

```
 cd dbxml-6.0.xx/xqilla
mkdir build
cd build
../configure --with-xerces=../../xerces-c-src
make
make install
```

This will install XQilla under `/usr/local`. If you want to install in a different directory, add `--prefix=/path/to/install` to the configure line. The --with-xerces directive is required, and must point to a valid installed Xerces tree.

To build multiple UNIX versions of XQilla in the same source tree, create a new directory at the same level as the build directory, and then configure and build in that directory as described previously.

Configuring and building Berkeley DB XML

The `--with-berkeleydb`, `--with-xerces`, and `--with-xqilla` configuration options can be used to specify the location of the install trees. Alternatively, the CFLAGS, LDFLAGS and other standard environment variables can be used to specify the location of the already installed include and library files.

To do a standard UNIX build of Berkeley DB XML using default paths and installations of the third-party libraries, change to the `build_unix` directory and then enter the following two commands, adjusting the paths as necessary:

```
cd dbxml-6.0.xx/dbxml/build_unix
../dist/configure
make
```

If you have changed the locations where Berkeley DB, Xerces, or XQilla are installed from the defaults, add the arguments `--with-berkeleydb=/path/to/db`, `--with-xqilla=/path/to/xqilla` and/or `--with-xerces=/path/to/xerces` to configure.

By default, Berkeley DB XML is installed in `/usr/local/BerkeleyDBXML.6.0`. To change that, add `--prefix=/path/to/install` to configure. To install the Berkeley DB XML library, enter the following command:

```
make install
```

To rebuild Berkeley DB XML, enter:

```
make clean
make
```

If you change your mind about how Berkeley DB XML is to be configured, you must start from scratch by entering the following command:

```
make distclean
../dist/configure
make
```

To build multiple UNIX versions of Berkeley DB XML in the same source tree, create a new directory at the same level as the build_unix directory, and then configure and build in that directory as described previously.

Building the Java API

To build the DB XML Java API, make sure there is a working `javac` in your `PATH`, and specify `--enable-java` when running both the Berkeley DB and Berkeley DB XML configure scripts. When you run `make`, the Java support library for Berkeley DB XML will be built, creating the file `dbxml.jar` in your build directory.

To make use of the DB XML Java API set your environment variable `CLASSPATH` to include the full pathname of the `dbxml.jar` file, as well as the `db.jar` file from Berkeley DB, and your environment variable `LD_LIBRARY_PATH` (or equivalent) to include the `.libs` subdirectory of your build directory.

Configuring Berkeley DB XML

There are several arguments you can specify when configuring Berkeley DB XML. Although only the Berkeley DB XML-specific ones are described here, most of the standard autoconf arguments are available and supported. To see a complete list of possible arguments, specify the --help flag to the configure program.

The Berkeley DB XML specific arguments are as follows:

- --enable-debug

  Use this flag to build with -g as a compiler flag and with DEBUG defined during compilation. This will create libraries and utilities with debugging symbols. This argument should not be specified when configuring to build production binaries.

- `--enable-java`

  To build the Berkeley DB XML Java API, enter --enable-java as an argument to configure. To build Java, you must also build with shared libraries. Before configuring, you must set your PATH environment variable to include javac. Note that it is not sufficient to include a symbolic link to javac in your PATH because the configuration process uses the location of javac to determine the location of the Java include files (for example, jni.h). On some systems, additional include directories may be needed to process jni.h; see Changing compilation and link options (page 12) for more options.

- `--enable-perl`

  Use this flag to build with Perl support. This builds Perl support for both Berkeley DB and Berkeley DB XML. This option requires a working perl program in your PATH.

- `--enable-shared`, `--enable-static`

  On systems supporting shared libraries, Berkeley DB XML builds both static and shared libraries by default. (Shared libraries are built using the GNU Project's Libtool distribution, which supports shared library builds on many (although not all) systems.) To not build shared libraries, configure using the --enable-shared='no' argument. To not build static libraries, configure using the --enable-static='no' argument.

- `--enable-test`

  To build the Berkeley DB XML test suite, enter --enable-test as an argument to configure. To run the Berkeley DB XML test suite, you must also build the Tcl version of the Berkeley DB XML library by specifying--with-tcl=path. If Java is enabled and you use --enable-test you will also be required to supply the --with-junit flag (see below).

- `--with-junit`

  Path to the JUnit jar file, for example --with-junit=. This option is required if both --enable-test and -enable-java are specified.

- `--with-berkeleydb=DIR`

  To specify the location of a Berkeley DB installation to be used when building Berkeley DB XML, enter --with-berkeleydb=DIR, replacing DIR with the path to the top-level directory of the Berkeley DB installation.

- `--with-xerces=DIR`

  To specify the location of a Xerces installation to be used when building Berkeley DB XML, enter --with-xerces=DIR, replacing DIR with the path to the top-level directory of the Xerces installation.

- `--with-xqilla=DIR`

  To specify the location of a XQilla installation to be used when building Berkeley DB XML, enter --with-xqilla=DIR, replacing DIR with the path to the top-level directory of the XQilla installation.

- `--with-tcl=DIR`

  To build the Tcl version of the Berkeley DB XML libraries, enter --with-tcl=DIR, replacing DIR with the directory in which the Tcl tclConfig.sh file may be found. See `Loading Berkeley DB with Tcl` in the *Berkeley DB Programmer's Reference Guide* for information on sites from which you can download Tcl and which Tcl versions are compatible with Berkeley DB XML. To build Tcl, you must also build with shared libraries.

Changing compilation and link options

You can specify compiler and/or compile and load time flags by using environment variables during Berkeley DB XML configuration. For example, if you want to use a specific compiler, specify the CC environment variable before running configure:

```
prompt: env CC=gcc ../dist/configure
```

Using anything other than the native compiler will almost certainly mean that you'll want to check the flags specified to the compiler and loader, too.

To specify debugging and optimization options for the C compiler, use the CFLAGS environment variable:

```
prompt: env CFLAGS=-O2 ../dist/configure
```

To specify header file search directories and other miscellaneous options for the C preprocessor and compiler, use the CPPFLAGS environment variable:

```
prompt: env CPPFLAGS=-I/usr/contrib/include ../dist/configure
```

To specify debugging and optimization options for the C++ compiler, use the CXXFLAGS environment variable:

```
prompt: env CXXFLAGS=-Woverloaded-virtual ../dist/configure
```

To specify miscellaneous options or additional library directories for the linker, use the LDFLAGS environment variable:

```
prompt: env LDFLAGS="-N32 -L/usr/local/lib" ../dist/configure
```

If you want to specify additional libraries, set the LIBS environment variable before running configure. For example, the following would specify two additional libraries to load, "posix" and "socket":

```
prompt: env LIBS="-lposix -lsocket" ../dist/configure
```

Make sure that you prepend -L to any library directory names and that you prepend -I to any include file directory names! Also, if the arguments you specify contain blank or tab characters, be sure to quote them as shown previously; that is with single or double quotes around the values you are specifying for LIBS.

The env command, which is available on most systems, simply sets one or more environment variables before running a command. If the env command is not available to you, you can set the environment variables in your shell before running configure. For example, in sh or ksh, you could do the following:

```
prompt: LIBS="-lposix -lsocket" ../dist/configure
```

In csh or tcsh, you could do the following:

```
prompt: setenv LIBS "-lposix -lsocket"
prompt: ../dist/configure
```

See your command shell's manual page for further information.

Installing Berkeley DB XML

Berkeley DB XML installs the following files into the following locations:

Configuration Variables	Default value
--prefix	/usr/local/BerkeleyDBXML.*Major*.*Minor*
--exec_prefix	$(prefix)
--bindir	$(exec_prefix)/bin
--includedir	$(prefix)/include
--libdir	$(exec_prefix)/lib
docdir	$(prefix)/docs

Berkeley DB XML uses the following default values for its installation locations:

Files	Default location
include files	$(includedir)
libraries	$(libdir)
utilities	$(bindir)
documentation	$(docdir)

There is one exception to the above values, and this follows the GNU Autoconf and GNU Coding Standards installation guidelines; please see that documentation for more information and rationale.

Also the Berkeley DB XML documentation is provided in HTML and PDF format, not in UNIX-style man or GNU info format. For this reason, Berkeley DB XML configuration does not support `--infodir` or `--mandir`. To change the default installation location for the Berkeley DB XML documentation, modify the Makefile variable, `docdir`.

When installing Berkeley DB XML on filesystems shared by machines of different architectures, please note that although Berkeley DB XML include files are installed based on the value of $(prefix), rather than $(exec_prefix), the Berkeley DB XML include files are not always architecture independent.

To move the entire installation tree to somewhere besides `/usr/local`, change the value of `prefix`.

To move the binaries and libraries to a different location, change the value of `exec_prefix`. The values of `includedir` and `libdir` may be similarly changed.

Any of these values except for `docdir` may be set as part of the configuration:

```
prompt: ../dist/configure --bindir=/usr/local/bin
```

Any of these values, including `docdir`, may be changed when doing the install itself:

```
prompt: make prefix=/usr/contrib/bdb install
```

The Berkeley DB XML installation process will attempt to create any directories that do not already exist on the system.

Dynamic shared libraries

Warning: the following information is intended to be generic and is likely to be correct for most UNIX systems. Unfortunately, dynamic shared libraries are not standard among UNIX systems, so there may be information here that is not correct for your system. If you have problems, consult your compiler and linker manual pages, or your system administrator.

The Berkeley DB XML dynamic shared libraries are created with the name libdbxml-`major.minor`.so, where `major` is the major version number and `minor` is the minor version number. Other shared libraries are created if Java and Tcl support are enabled: specifically, libdbxml_java-`major.minor`.so and libdbxml_tcl-`major.minor`.so.

On most UNIX systems, when any shared library is created, the linker stamps it with a "SONAME". In the case of Berkeley DB XML, the SONAME is libdbxml-`major.minor`.so. It is important to realize that applications linked against a shared library remember the SONAMEs of the libraries they use and not the underlying names in the filesystem.

When the Berkeley DB XML shared library is installed, links are created in the install lib directory so that libdbxml-`major.minor`.so, libdbxml-`major`.so, and libdbxml.so all refer to the same library. This library will have an SONAME of libdbxml-`major.minor`.so.

Any previous versions of the Berkeley DB XML libraries that are present in the install directory (such as libdbxml-2.7.so or libdbxml-2.so) are left unchanged. (Removing or moving old shared libraries is one drastic way to identify applications that have been linked against those vintage releases.)

Once you have installed the Berkeley DB XML libraries, unless they are installed in a directory where the linker normally looks for shared libraries, you will need to specify the installation directory as part of compiling and linking against Berkeley DB XML. Consult your system manuals or system administrator for ways to specify a shared library directory when compiling and linking applications with the Berkeley DB XML libraries. Many systems support environment variables (for example, LD_LIBRARY_PATH or LD_RUN_PATH), or system configuration files (for example, /etc/ld.so.conf) for this purpose.

Warning: some UNIX installations may have an already existing `/usr/lib/libdbxml.so`, and this library may be an incompatible version of Berkeley DB XML. They also may have installed, incompatible versions of Berkeley DB (libdb*.so), Xerces-C (libxerces*.so) and XQilla (libxqilla*.so). Odd errors can result from mixing/matching such versions.

We recommend that applications link against libdbxml.so (for example, using -ldbxml). Even though the linker uses the file named libdbxml.so, the executable file for the application remembers the library's SONAME (libdbxml-`major.minor`.so). This has the effect of marking the applications with the versions they need at link time. Because applications locate their needed SONAMEs when they are executed, all previously linked applications will continue to run using the library they were linked with, even when a new version of Berkeley DB XML is installed and the file `libdbxml.so` is replaced with a new version.

Applications that know they are using features specific to a particular Berkeley DB XML release can be linked to that release. For example, an application wanting to link to Berkeley DB XML major release "3" can link using -ldbxml-3, and applications that know about a particular minor release number can specify both major and minor release numbers; for example, -ldbxml-3.5.

If you want to link with Berkeley DB XML before performing library installation, the "make" command will have created a shared library object in the `.libs` subdirectory of the build directory, such as `build_unix/.libs/libdbxml-major.minor.so`. If you want to link a file against this library, with, for example, a major number of "3" and a minor number of "5", you should be able to do something like the following:

```
cc -L BUILD_DIRECTORY/.libs -o testprog testprog.o -ldbxml-3.5
env LD_LIBRARY_PATH="BD/.libs:$LD_LIBRARY_PATH" ./testprog
```

where `BD` is the full directory path to the directory where you built Berkeley DB XML.

The libtool program (which is configured in the build directory) can be used to set the shared library path and run a program. For example, the following runs the gdb debugger on the db_dump utility after setting the appropriate paths:

```
libtool gdb db_dump
```

Libtool may not know what to do with arbitrary commands (it is hardwired to recognize "gdb" and some other commands). If it complains the mode argument will usually resolve the problem:

```
libtool --mode=execute my_debugger db_dump
```

On most systems, using libtool in this way is exactly equivalent to setting the LD_LIBRARY_PATH environment variable and then executing the program. On other systems, using libtool has the virtue of knowing about any other details on systems that don't behave in this typical way.

Building and Using Berkeley DB XML Applications on UNIX/POSIX Systems

Any application which uses Berkeley DB XML, regardless of language API used, relies on four libraries: Berkeley DB XML, Berkeley DB, XQilla and Xerces C++. When compiling, the include

files from these packages must be available, and the libraries must be included when linking the application.

When running an application, each of the shared libraries must be located by the dynamic linker, usually controlled by system settings and the LD_LIBRARY_PATH environment variable. This applies to both C++ and Java applications. See Dynamic shared libraries (page 14) for more information

Building C++ Applications

Assuming Berkeley DB XML, Berkeley DB, Xerces C++ and XQilla have all been built using buildall.sh and installed in the default location, a typical command to compile C++ code that uses Berkeley DB XML is:

```
g++ -c -Ipath_to_distribution/install/include -c myapp.cpp
```

If the libraries are in different locations, such as Xerces and XQilla in /usr/local, Berkeley DB in /usr/local/BerkeleyDB.6.1, and Berkeley DB XML in /usr/local/BerkeleyDBXML.6.0, a command might be:

```
g++ -c -I/usr/local/include
  -I/usr/local/BerkeleyDBXML.6.1/include
  -I/usr/local/BerkeleyDB.6.0/include -c myapp.cpp
```

Then, corresponding link commands are:

```
g++ -o myapp myapp.o -Lpath_to_distribution/install/lib
  -lxqilla -lxerces-c -ldbxml-6.0 -ldb-6.1
```

and

```
g++ -o myapp myapp.o -L/usr/local/lib
  -L/usr/local/BerkeleyDBXML.6.0/lib
  -L/usr/local/BerkeleyDB.6.1/lib -lxqilla
  -l xerces-c -ldbxml-6.0 -ldb-6.1
```

Building Java Applications

Java applications compile normally using javac and require both db.jar and dbxml.jar in their CLASSPATH. Assuming buildall.sh and default installation directories were used these reside in the directory p2d/install/lib. This is a reasonable javac line:

```
javac -cp"p2d/install/lib/db.jar:p2d/install/lib/dbxml.jar" \
MyClass.java
```

where p2d is the path to your distribution.

In order to run the resulting program both CLASSPATH and LD_LIBRARY_PATH (or equivalent) must be set properly. For example:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:p2d/install/lib
export CLASSPATH=$CLASSPATH:\
p2d/install/lib/db.jar:p2d/install/lib/dbxml.jar
```

```
java MyClass
```

Architecture Independent Build FAQ

1. I cannot un-tar the Berkeley DB XML product archive.

   The Berkeley DB XML archive includes some paths that are longer than some tar programs can handle (for example, AIX, and Solaris). In general, the required files are unpacked, and the limitation will not be noticed. The files in question are not in the main product, and will not affect the build and installation.

   If there is an error during tar, or a filename appears to have been truncated, you can either install and use a different tar program, such as GNU tar or use the Windows distribution (a .zip archive). If you have another machine on your network that has GNU tar, you can use that machine to do the work, via NFS.

AIX Build Notes

See also Berkeley DB notes on AIX

1. Normal build command for AIX using xlC_r and xlc_r and no options:

   ```
   sh buildall.sh -x xlC_r -c xlc_r
   ```

2. I get a link error regarding open and open64 on AIX 5.3 while linking against Berkeley DB.

   On AIX 5.3, you may see a link failure while building Berkeley DB XML against Berkeley DB 6.1.x with "open" or "open64" in the message. This may occur when trying to link dbxml_dump or dbxml_load.

   This can be worked around by adding this line to db-6.1.x/build_unix/db.h:

   ```
   #include <fcntl.h>
   ```

   It should be added near the top of the file, before the include of db.h. After this, rebuild Berkeley DB by changing to the directory db-6.1.x/build_unix and running "make clean; make; make install."

3. I get an error regarding truncate64 and stat64 on AIX 5.3 while building Berkeley DB.

   On AIX 5.3, you may see a compilation failure while building Berkeley DB 4.3.29 of the nature, "...truncate64 is not a member of..." or "...stat64 is not a member of..."

   This can be worked around by adding this line to db-4.3.29/dbinc/db.in:

   ```
   #include <unistd.h>
   ```

   It should be added just after the line:

   ```
   #ifndef __NO_SYSTEM_INCLUDES.
   ```

   After this, re-run the buildall.sh script, and be sure that Berkeley DB configures itself again, regnerating the file, db-4.3.29/build_unix/db.h.

Solaris Build Notes

See also Berkeley DB notes on Solaris

1. Solaris build commands

   Normal build command for Solaris using Sun Workshop 64-bit and no options:

   ```
   bash ./buildall.sh -c cc -x CC \
       --with-configure-env="CFLAGS='-xarch=v9' \
       CXXFLAGS='-xarch=v9' LDFLAGS='-xarch=v9'"
   ```

   Normal build command for Solaris using Gnu tools and no options, 32-bit:

   ```
   bash ./buildall.sh -c gcc -x g++ \
       --with-configure-env="CFLAGS='-mcpu=v9' \
       CXXFLAGS='-mcpu=v9' LDFLAGS='-mcpu=v9 \
       -L$PWD/install/lib'"
   ```

   Normal build command for Solaris using Gnu tools and no options, 64-bit:

   ```
   export LD_LIBRARY_PATH=/usr/sfw/lib/64
   bash ./buildall.sh -c gcc -x g++ --with-configure-env="CFLAGS='-m64 \
       -mcpu=v9' CXXFLAGS='-m64 -mcpu=v9 -L$PWD/install/lib' \
       LDFLAGS='-m64 -mcpu=v9 -L$PWD/install/lib
   ```

   Some Solaris system can't build debug library with -g option, it is a known issue in sun's C ++ library. Use -g0 instead.

   When building using Sun Workshop there may be a link error indicating that the symbol "pow" or "sqrt" is undefined. If this occurs add "-lm" to LIBS in the build command.

FreeBSD Build Notes

See also Berkeley DB notes on FreeBSD

1. Normal build command for FreeBSD and no options:

   ```
   sh buildall.sh
   ```

2. Known issue in Xerces 3.0.1.

   There is a known issue in xerces 3.0.1 where in the file xerces-c-src/src/xercesc/util/NetAccessors/Socket/UnixHTTPURLInputStream.cpp the "if (n<0)" should be "if(n!=0)".

3. I can't run Berkeley DB XML on a FreeBSD 5.4 machine.

   On some installations of FreeBSD 5.4, you may see this runtime error:

   ```
   Fatal error 'Spinlock called when not threaded.' at line 87 in file
   /usr/src/lib/libpthread/thread/thr_spinlock.c (errno = 0)
   ```

   If this error is seen, it is necessary to create an /etc/libmap.conf file to map libc_r to libpthread. See "man 4 libmap.conf" for details on how to do this.

Mac OS X Build Notes

See also Berkeley DB notes on Mac OS X

1. Normal build command for OS X and no options:

```
sh buildall.sh
```

To build debug using the dwarf format (quite useful):

```
bash buildall.sh --with-configure-env="CXXFLAGS='-g -gdwarf-2'" \
    --with-configure-env="CFLAGS='-g -gdwarf-2'" --enable-debug
```

2. When trying to link multiple Berkeley DB XML language interfaces (for example, Tcl, C++, Java, Python) into a single process, I get "multiple definitions" errors from dyld.

This problem will most often occur with Python, PHP, or Tcl, where libraries are dynamically loaded. To fix this problem, set the environment variable MACOSX_DEPLOYMENT_TARGET to 10.3 (or your current version of OS X), and reconfigure and rebuild Berkeley DB XML from scratch. See the OS X ld(1) and dyld(1) man pages for information about how OS X handles symbol namespaces, as well as undefined and multiply-defined symbols.

Cygwin Build Notes

1. Cygwin build commands

```
bash buildall.sh -c gcc -x g++ --with-configure-env="LDFLAGS='\
    -no-undefined'"
```

2. Building dbxml-tcl lib on cygwin

Install tcl libs on cygwin by cygwin setup application.

Modify db-6.1.x/dist/Makefile.in:

```
- LIBTSO_LIBS=    @LIBTSO_LIBS@ @LIBSO_LIBS@
+ LIBTSO_LIBS=    @LIBTSO_LIBS@ @LIBSO_LIBS@ -ltcl
```

Modify dbxml/Makefile.in:

```
- libdbxml_tcl_la_LDFLAGS = -release $(DBXML_VERSION_MAJOR).$(DBXML_
VERSION_MINOR) $(TCL_LIBRARY_EXTENSION)
+ libdbxml_tcl_la_LDFLAGS = -release $(DBXML_VERSION_MAJOR).$(DBXML_
VERSION_MINOR) $(TCL_LIBRARY_EXTENSION) -no-undefined -ltcl -ltclstub
```

Build dbxml:

```
./buildall.sh --enable-test --with-tcl=/usr/lib/ \
--with-configure-env="LDFLAGS='-no-undefined'"
```

Now the static libs of dbxml_tcl are generated. If you want dbxml_tcl.dll, continue to next step:

Build dbxml_tcl-6-0.dll

```
cd dbxml/build_unix
g++ -shared -nostdlib .libs/libdbxml_tcl_la-dbxml_tcl_wrap.o \
-ldbxml -o ../../install/bin/cygdbxml_tcl-6-0.dll -lstdc++ \
-ltcl -lgcc -lcygwin -lkernel32 -ldb_tcl-6.1 \
-L../../install/lib -L.libs/

cd ../../install/bin

ln -sf cygdb_tcl-6.1.dll libdb_tcl-6.1.dll

ln -sf cygdbxml_tcl-6-0.dll libdbxml_tcl-6-0.dll
```

Chapter 3. Building Berkeley DB XML for Windows

If you want to build on Windows using Cygwin, see Building Berkeley DB XML for UNIX/POSIX systems (page 4) and Cygwin Build Notes (page 19).

The Berkeley DB XML distribution comprises several libraries: a base C++ library, three third-party libraries, and optional libraries for additional language interfaces such as Java, Perl, Python, PHP, and Tcl. Instructions for building the base libraries as well as Java, Tcl and PHP interfaces are included here. Instructions for the other language bindings appear in their respective directories:

```
dbxml-6.0.xx/dbxml/src/{perl,python}.
```

All bindings require the C++ library and third-party libraries.

Third Party Libraries

Berkeley DB XML makes use of several open source libraries. Each of these packages is freely available and distributed under an Open Source license. The Berkeley DB XML distribution bundles compatible versions of all third-party libraries. No additional downloads are required. Use of other versions may not work. If in doubt ask on the Berkeley DB XML forum. Berkeley DB XML requires the following third-party libraries:

- Berkeley DB

  Berkeley DB is a general purpose database toolkit. This release of Berkeley DB XML includes the Berkeley DB 6.1.x release, and requires the Berkeley DB 4.3.28 release or later.

- Xerces

  Xerces is the Apache implementation of XML parsing technology and DOM. The build requires a source release of Xerces, not just a binary release. This release of Berkeley DB XML bundles Xerces-C 3.1.2 and requires the Xerces 3.0.1 release or newer.

- XQilla

  XQilla is an open source partial implementation of XQuery 3.0 and XPath 2.0. Berkeley DB XML bundles the 2.3 release of XQilla. Later versions of XQilla may work, but are not implicitly supported.

- ZLIB

  ZLIB is an open source compression library. This release of Berkeley DB XML bundles the binary release of ZLIB 1.2.3. It is not required if wholedoc container compression is not desired.

Building with Microsoft Visual Studio 2010 and 2012

- Choose *File -> Open -> Project/Solution*. Look in the `dbxml-6.0.xx/dbxml/build_windows` directory for solution files, select `BDBXML_all_vs2010.sln`, and press Open.

- If using Visual Studio 2012, choose to upgrade the solution and projects.
- Choose the project configuration from the drop-down menu on the Visual Studio tool bar.
- To build, select *Build Solution* from the *Build* drop-down menu. All library files (*.lib) are placed in `dbxml-6.0.xx/lib`, DLLs and executables are installed in `dbxml-6.0.xx/bin` or `dbxml-6.0.xx/bin/debug` depending on the configuration you choose, .jar files are placed in `dbxml-6.0.xx/jar`, and header (include) files for application development are copied to `dbxml-6.0.xx/include`.
- By default, `BDBXML_all_vs2010.sln` builds all third-party libraries, Berkeley DB XML, and Berkeley DB XML C++ examples. The examples are installed in `bin{/debug}` along with other executables.

Building with Microsoft Visual Studio 2008 and Express Edition

- Choose *File* -> *Open* -> *Project/Solution*. Look in the `dbxml-6.0.xx/dbxml/build_windows` directory for solution files, select `BDBXML_all.sln`, and press Open. When given the option to upgrade the solution and projects, choose yes for all.
- Choose the project configuration from the drop-down menu on the Visual Studio tool bar.
- To build, select *Build Solution* from the *Build* drop-down menu. All library files (*.lib) are placed in `dbxml-6.0.xx/lib`, DLLs and executables are installed in `dbxml-6.0.xx/bin` or `dbxml-6.0.xx/bin/debug` depending on the configuration you choose, .jar files are placed in `dbxml-6.0.xx/jar`, and header (include) files for application development are copied to `dbxml-6.0.xx/include`.
- By default, `BDBXML_all.sln` builds all third-party libraries, Berkeley DB XML, and Berkeley DB XML C++ examples. The examples are installed in `bin{/debug}` along with other executables.
- Note that Visual Studio 2008 Express Edition does not include an x64 compiler, and other versions of Visual Studio 2008 do not install the x64 compiler and tools by default.

Building with Microsoft Visual Studio 2005

- Choose *File* -> *Open* -> *Project/Solution*. Look in the `dbxml-6.0.xx/dbxml/build_windows` directory for solution files, select `BDBXML_all.sln`, and press Open.
- Choose the project configuration from the drop-down menu on the Visual Studio tool bar.
- To build, select *Build Solution* from the *Build* drop-down menu. All library files (*.lib) are placed in `dbxml-6.0.xx/lib`, DLLs and executables are installed in `dbxml-6.0.xx/bin` or `dbxml-6.0.xx/bin/debug` depending on the configuration you choose, .jar files are placed in `dbxml-6.0.xx/jar`, and header (include) files for application development are copied to `dbxml-6.0.xx/include`.
- By default, `BDBXML_all.sln` builds all third-party libraries, Berkeley DB XML, and Berkeley DB XML C++ examples. The examples are installed in `bin{/debug}` along with other executables.

Building with Microsoft Visual C++ 2005 Express Edition

- Download and install Microsoft Platform SDK.
- Choose *File -> Open -> Project/Solution*. Look in the `dbxml-6.0.xx/dbxml/build_windows` directory for solution files, select `BDBXML_all.sln`, and press Open.
- Right click the `db` project and select *properties*. On the properties page select *All Configurations* from the *Configuration menu* in the top left corner. Enter `ws2_32.lib advapi32.lib` for the `Additional Dependencies` option of *Linker -> Input*. Select the OK button to save the changes.
- Select *Options...* from the *Tools* pull-down menu. Choose *Projects and Solutions*, then *VC++ Directories*. Select *Include files* from the *Show directories for* menu. Add the complete path to `Microsoft Platform SDK\Include\mfc` and `Microsoft Platform SDK\Include` to the list of directories. Next select *Library files* from the *Show directories for:* menu. Add the complete path to `Microsoft Platform SDK\Lib` to the list of directories.
- Choose the project configuration from the drop-down menu on the Visual Studio tool bar.
- To build, select *Build Solution* from the *Build* drop-down menu. All library files (*.lib) are placed in `dbxml-6.0.xx/lib`, DLLs and executables are installed in `dbxml-6.0.xx/bin` or `dbxml-6.0.xx/bin/debug` depending on the configuration you choose, .jar files are placed in `dbxml-6.0.xx/jar`, and header (include) files for application development are copied to `dbxml-6.0.xx/include`.
- By default, `BDBXML_all.sln` builds all third-party libraries, Berkeley DB XML, and Berkeley DB XML C++ examples. The examples are installed in `bin{/debug}` along with other executables.

Running C++ Examples

After a successful build, the Berkeley DB XML example executable files are in `bin{/debug}`. See `dbxml/examples/cxx/gettingStarted/Readme.txt` for instructions on building the C++ example containers and running examples.

Building the Java API with Visual Studio

The Berkeley DB XML Java API is not built automatically. The following instructions assume that you have installed the Sun Java Development Kit in `d:/java`. If you installed elsewhere or have different Java software you will need to adjust the pathnames accordingly.

1. Set your include directories. Choose *Tools -> Options -> Projects -> VC++ Directories*. Under the "Show directories for" pull-down, select "Include files". Add the full pathnames for the `java/include` and `java/include/win32` directories. Then click OK. These are the directories needed when including jni.h.
2. Set your executable files directories. Choose *Tools -> Options -> Projects -> VC++ Directories*. Under the "Show directories for" pull-down, select "Executable files". Add the

full pathname for the `java/bin` directory, then click OK. This is the directory needed to find javac.

3. Set the build type to Release or Debug (or other appropriate configuration) in the drop-down on the .NET tool bar.

4. Right-click on db_java and select Build. This builds the Java support library for Berkeley DB, which is required for Berkeley DB XML, and compiles all the java files, placing the resulting `db.jar` and `dbexamples.jar` files in `bin` or `bin/debug`.

5. Right-click on dbxml_java and select Build. This builds the Java support library for Berkeley DB XML, and compiles all the java files, placing the resulting `dbxml.jar` and `dbxmlexamples.jar` files in `jar`.

Building the Tcl API

Tcl support is not built automatically. You must have Tcl installed on the machine. You should use at least version 8.5. These notes assume that Tcl is installed as `d:/tcl` but you can change that if you want.

The Tcl library must be built as the same build type as the Berkeley DB and Berkeley DB XML libraries (both Release or both Debug). We found that the binary release of Tcl can be used with the Release configurations of Berkeley DB and Berkeley DB XML, but you will need to build Tcl from sources for the Debug configuration. Before building Tcl, you will need to modify its makefile to make sure that you are building a debug version, including thread support. This is because the set of DLLs linked into the Tcl executable must match the corresponding set of DLLs used by Berkeley DB and Berkeley DB XML.

Building Tcl with Visual Studio

1. Set the include directories. Choose *Tools* -> *Options* -> *Projects* -> *VC++ Directories*. Under the "Show directories for" pull-down, select "Include files". Add the full pathname for `tcl/include` , then click OK. Th is the directory that contains tcl.h.

2. Set library files directory. Choose *Tools* -> *Options* -> *Projects* -> *VC++ Directories*. Under the "Show directories for" pull-down, select "Library files". Add the full pathname for the `tcl/lib` directory, then click OK. This is the directory needed to find `tcl85.lib` (or whatever the library is named in your distribution).

3. Set the build type to Release or Debug (or other appropriate configuration) in the drop-down on the .NET tool bar.

4. Right-click on db_tcl and select Build. This builds the TCL support library for Berkeley DB, which is required for Berkeley DB XML, and compiles all the files, placing the result into `bin/debug/libdb_tcl61d.dll` or `bin/libdb_tcl61.dll`.

5. Right-click on dbxml_tcl and select Build. This builds the Tcl support library for Berkeley DB XML, placing the result into `bin/debug/libdbxml_tcl60d.dll` or `bin/libdbxml_tcl60.dll`.

If you use a version different from Tcl 8.5.x you will need to change the name of the Tcl library used in the build (for example, `tcl85.lib`) to the appropriate name. To do this, right click on *db_tcl*, go to *Properties -> Linker -> Input -> Additional dependencies* and change `tcl85.lib` to match the Tcl version you are using.

Disabling default compression

Disabling default compression has the benefit of speeding up access to data in whole document containers at the cost of increasing container size. Disabling default compression also removes the dependency on the ZLIB library. Compression can be disabled by deleting the preprocessor definition `_DBXML_COMPRESSSION` and the linker dependency zdll.lib from the project dbxml, then rebuilding the project. Be warned that containers created with default compression cannot be opened if default compression has been disabled.

Using Berkeley DB XML on Windows

An application which uses Berkeley DB XML relies on four libraries: Berkeley DB XML, Berkeley DB, XQilla, and Xerces C++. When compiling, some include files from these packages must be available, and the libraries must be included when linking the application.

When running the application, each of the DLLs must be available in a directory in the `PATH`. You can achieve this either by copying all of the DLLs into a directory that is already in your `PATH`, such as `windows/system32`, or by adding each of the directories containing one of the DLLs to your `PATH` variable. The default build of Berkeley DB XML places all library DLLs into a single directory to simplify this process.

Berkeley DB XML Include File Locations

A typical Berkeley DB XML application requires only include files from Berkeley DB XML and Berkeley DB. If an application uses interfaces from the Xerces C++ library, it must include Xerces header files as well.

The following are include file locations you may need for the Berkeley DB XML distribution. The include files are copied during the build. If you have a successful build, or you have the binary distribution, only the first location is necessary. Version numbers may need to be modified to match your distribution:

```
dbxml-6.0.xx/include
dbxml-6.0.xx/dbxml/include
dbxml-6.0.xx/db-6.1.26/build_windows
dbxml-6.0.xx/xerces-c-src/src
```

Berkeley DB XML Include File Locations

The Berkeley DB XML build places libraries against which applications must link in `dbxml-6.0.xx/lib`. This directory holds both Release and Debug .lib files. An application that uses only Berkeley DB XML interfaces is only required to link with the Berkeley DB XML library. If an application uses Berkeley DB or Xerces C++ interfaces, it is necessary to link with those libraries as well.

The following are the important library filenames for the Berkeley DB XML distribution (version numbers may need to be modified to match your distribution):

Release Build	Debug Build
libdbxml60.lib	libdbxml60d.lib
libdb61.lib	libdb61d.lib
xerces-c_3.lib	xerces-c_3D.lib

Berkeley DB XML DLL Names and Locations

The Berkeley DB XML build process places DLLs and executable files in `dbxml-6.0.xx/bin` for the Release build, and `dbxml-6.0.xx/bin/debug` for the Debug build. Unlike header files and .lib files, all library DLLs must be available at run time in the application's `PATH` environment variable.

The following are the DLLs required for Berkeley DB XML applications (version numbers may need to be modified to match your distribution):

Release Build	Debug Build
libdbxml60.dll	libdbxml60d.dll
libdb61.dll	libdb61d.dll
xerces-c_3.dll	xerces-c_3D.dll
xqilla23.dll	xqilla23d.dll

A good way to ensure that your application uses the necessary header files and libraries is to use one of the Berkeley DB XML example projects (`dbxml/build_windows/dbxml_example*`) as a template for your application's project file.

Berkeley DB XML Binaries on Windows

Executable files and libraries are available for Berkeley DB XML. They may be downloaded in the form of a Windows installer package `dbxml-6.0.xx.msi` which must then be installed. The binaries are built using Microsoft Visual Studio 2008 and include language bindings for C++, Java, Python, and Perl. Java is compiled against Java 7, Python is compiled against the 2.7 release of ActivePython, Perl is compiled using ActivePerl 5.16.

There are several installation options for installing Windows binaries ranging from a simple runtime-only deployment to a development installation for all languages. Source code other than examples is not available in the Windows installer package. If source code is desired one of the source downloads must be used.

Windows Installer Options

When installing Berkeley DB XML using the Windows installer you can choose to install any or all of a number of options. By default all options are selected.

- `Core Runtime`

This option includes only the .dll and .exe files required to run Berkeley DB XML C++ applications. This option can only be used to run already-built applications; it cannot be used to build new C++ applications. Program Database (.pdb) files are included for executables and libraries. This option includes the PHP runtime libraries.

- `C++ Development`

  This option bundles the runtime libraries and adds the header files and libraries (.lib files) required to compile and link new C++ applications.

- `Documentation`

  This option installs the Berkeley DB XML documentation set.

- `Examples`

  This option installs example programs for all language bindings.

- `Java Runtime`

  This option installs the Java runtime files (.jar) along with the core runtime (native) libraries which are also required for Berkeley DB XML Java applications.

- `Python Runtime`

  This option installs executables for Python that require further installation for use. See instructions below.

- `Perl Runtime`

  This option installs the core runtime and Perl binaries which require further installation for use. See instructions below.

Installation Layout

The Windows installer allows a choice of installation location. All paths used below are relative to that directory. The installation layout is the same as it is for the default build from source:

- `bin`

  Holds exectuables for the core runtime

- `lib`

  Holds libraries (.lib) for the C++ development option

- `include`

  Holds header files for the C++ development option

- `jar`

Holds Java jar files for the Java runtime option

- `dbxml/doc`

  Holds Berkeley DB XML examples for all languages

- `perl`

  Holds Perl binaries

- `python`

  Holds Python binaries

General Note on Using Windows Binaries

Because of the nature of some of the scripting language integrations it can be necessary to use the same compiler for the scripting language itself as for language extensions such as Berkeley DB XML support. For this reason the binaries provided may not work in all cases. If a particular installation has problems it may be necessary to compile from source in the case of Python and Perl.

Using C++ Runtime Libraries and Programs

It is necessary to set your PATH to run all Berkeley DB XML C++ and Java applications PATH needs `<InstallDir>/bin`. By default the installer will set the PATH and CLASSPATH of the installing user to include the relevant directories and files.

C++ example programs, if installed, can be found in `<InstallDir>/dbxml/examples/cxx`. Visual Studio project files for the examples are in `<InstallDir>/dbxml/build_windows` and can be used as templates for new applications.

Using Java Binaries

It is necessary to set your PATH and CLASSPATH to run Berkeley DB XML Java applications. CLASSPATH needs `<InstallDir>/jar/db.jar;<InstallDir>/jar/dbxml.jar`. See above for PATH additions. By default the installer will set the PATH and CLASSPATH of the installing user to include the relevant directories and files.

Java example programs, if installed, can be found in `<InstallDir>/dbxml/examples/java`. Many users develop and deploy Berkeley DB XML Java applications in an application server environment such as Tomcat. Most of these servers ignore PATH and CLASSPATH for applications (for security reasons) so it is necessary to follow product-specific instructions for correct application deployment.

Installing and Using Python Binaries

Python binaries are built and packaged in executable files that can be installed on a machine that has the 32 bit ActivePython 2.7 already installed. There are two files that need to be installed:

1. `<InstallDir>/python/bsddb3-6.1.#.win32-py2.7.exe`

   Execute this to install pybsddb release 6.1.

2. `><InstallDir>/python/dbxml-6.0.xx.win32-py2.7.exe`

   Execute this to install Berkeley DB XML libraries

The executables above include `all` of the dependent libraries so it is not necessary to install either of the core runtime libraries. The installation will fail if Python 2.7 is not installed on the target machine.

Python example programs, if installed, can be found in `<InstallDir>/dbxml/examples/python`. A simple smoke test that can be done after installation is running the Python example script `<InstallDir>/dbxml/examples/python/basic/helloWorld.py`. If this works then your installation is good.

Installing and Using Perl Binaries

Perl binaries are built and bundled into the compressed tar archive file `<InstallDir>/perl/dbxml_perl.tar.gz`. There is also an ActivePerl PPD file — `<InstallDir>/perl/dbxml_perl.ppd` -- which can be used by the PPM program for installation. If you have 32 bit ActivePerl version 5.16 or later installed on your system and it includes PPM, then you can do this to install the Perl binaries after initial installation:

- change directory to `<InstallDir>/perl`
- type "ppm install dbxml_perl.ppd"

This will install the Berkeley DB XML Perl binary files in the appropriate location for your system.

Perl example programs, if installed, can be found in `<InstallDir>/dbxml/examples/perl/gettingStarted`. When using these examples, always start with `loadExamplesData.pl`. There is a Readme.txt file for the gettingStarted examples in this directory.

Troubleshooting on Windows

Compiling Errors

Compiling fails with the message "Error, cannot find include file windows.h" or "cannot open include file 'winres.h'".

That problem happens because the file, which is needed to compile programs that make calls to the Windows operating system, is not installed. To fix this, download and install the Microsoft Windows SDK for your system (it is free).

Once the SDK is installed, add the file paths to Visual Studio. Select *Options...* from the *Tools* pull-down menu. Choose *Projects and Solutions -> VC++ Directories -> Show directories -> Include files*. Add the complete path to `Microsoft Platform SDK\Include\mfc` and

`Microsoft Platform SDK\Include` to the list of directories. Next select *Show directories -> Library files* Add the complete path to `Microsoft Platform SDK\Lib` to the list of directories. These instructions my vary depending on your SDK and Visual Studio version.

Linking Errors

Linking fails with the error "unsatisfied link error".

This error can have multiple causes, and can mean that the library in question is missing or not in the `PATH`, or that one of the dependent libraries is missing or cannot be loaded properly. A useful tool in debugging this error is the free program depends.exe.

If depends.exe shows that the missing libraries do not have a name like MSVC*.dll, and the files are on the computer, then directories containing the libraries must be added to the `PATH` environment variable. To do this add the directories to *Control Panel -> System -> Advance -> Environment Variables*.

Unable to find MSVC*.dll when linking, or program failed to execute because it could not find MSVC*.dll

If the missing files have names like MSVC*, then the situation is a bit more complicated. These are the standard template libraries of C++ and C, and they have to be linked through a manifest that is embedded in the libraries. Simply moving the libraries into the `PATH` will not fix the problem. This problem can have several causes and solutions.

1. Berkeley DB XML was compiled from source on a hard drive of type FAT.

   Visual Studio has a bug were the manifest does not update properly on FAT hard drives. To fix this modify the Visual Studio projects xerces, dbxml, db, db_java, dbxml_java, and xqilla as follows: Right click the project and select *Properties -> Configuration Properties -> Manifest Tool -> General* and change Use FAT32 Work-around to Yes. Then rebuild all of the projects.

2. Berkeley DB XML was compiled from source on a hard drive that is not of type FAT.

   In this case the files might not be installed on the computer. To fix this, download and install the Microsoft SDK as described in the `Compile Errors` section above.

3. The Berkeley DB XML binaries are installed.

   In this case the files are installed by the binary installation, but some dependent operating system files may be missing. In that case download and install Microsoft Visual C++ 2005 or 2008 SP1 Redistributable Package (x86 or X64, depending on the system, although the binaries are all 32 bit, so the x86 version may be required even if the system is different).

Execution Errors

Running Java results in an "Internal Error" when trying to open an `XmlManager` or Environment.

This has been reported as being caused by the Java cache being too large (especially on 64-bit systems running Java in 32-bit mode). The solutions is to reduce the size of the Java cache to below its maximum (which has been reported as being 486 MB, although this may very with different JVM) by starting Java with the flags `-Xms32m -Xmx300m`.

The Berkeley DB XML binaries are installed and running the dbxml shell, or any other program, results in the error message "Program failed to initialized due to error X(######)".

Missing files cause this. To solve this download Microsoft Visual C++ 2005 (or 2008) SP1 Redistributable Package (x86 or X64, depending on the system, although the binaries are all 32 bit, so the x86 version may be required even if the system is different).

Running Java from the Berkeley DB XML binaries on 64-bit Windows

This can be done, but it requires that the 64-bit JVM be run in 32-bit mode and with the flags `-Xms32m -Xmx300m`. To run in 64-bit mode requires that Berkeley DB XML be compiled from source as a 64-bit library.

Chapter 4. Berkeley DB XML XQuery Extension Functions

Berkeley DB XML supports the XQuery Specification from the W3C. The specification includes a large number of functions, and also supports the writing of extension functions. Berkeley DB XML includes a number of extension functions that are in the Berkeley DB XML namespace, `http://www.sleepycat.com/2002/dbxml`, which is assumed to be bound to the prefix "dbxml" in this documentation.

The functions signatures follow that of the specification. In addition to documenting the Berkeley DB XML XQuery extension functions, the treatment of the arguments to fn:collection() and fn:doc() in the context of Berkeley DB XML.

This documentation covers only those areas where Berkeley DB XML extends the specification or has implementation-defined behavior. See the XQuery Specification or other resources as XQuery references. The following functions are interpreted and implemented in Berkeley DB XML:

fn:collection()

```
fn:collection($arg as xs:string?) as node()*
```

Returns a sequence of nodes obtained by interpreting $arg as an xs:anyURI. If $arg is not specified, the default collection is used if it is set. See the specification for details.

Parameters

$arg

A URI representing the collection.

URI Resolution in Berkeley DB XML

A URI representing an Berkeley DB XML container is of the form, "dbxml:/container_alias," where "container_alias" is an absolute or relative path to the container, or an alias added using `XmlContainer::addAlias()`. If the URI is either of the form "dbxml:////absolute_path_to_container" or "dbxml:/C:/absolute_windows_path_to_container" (Windows) then the path is considered absolute; otherwise, it is relative to the Berkeley DB environment's data directory. In general, it's simplest to use an explicit alias. It is also important to note that the Berkeley DB XML query optimizer uses the argument to fn:collection() to determine which indices may be appropriate for a given query.

By default, the base URI in an `XmlQueryContext` is "dbxml:/" which allows use of "collection('relative_path_to_container')" without any additional code. It is possible to use `XmlQueryContext::setBaseURI()` to change the default base URI. Note also that the Windows pathname separator, '\', is not a valid separator in a URI; the forward slash ('/') must be used, even on Windows.

Some applications may choose to implement their own name resolution scheme for the argument to fn:collection(). This is supported by allowing applications to control the

resolution process using an application-provided instance of the `XmlResolver` class, and using a URI that looks like (for example) "myscheme:/my_name_for_container." Collections located in this manner cannot currently be optimized by indices.

dbxml:contains()

```
dbxml:contains($toLookIn as xs:string?, $toLookFor as xs:string?) as
xs:boolean
```

This function behaves identically to fn:contains() in the XQuery 1.0 and XPath 2.0 Functions and Operators Specification except that it operates in a case- and diacritic-insensitive manner. This allows case-insensitive queries that can be optimized by Berkeley DB XML substring indices.

fn:doc()

```
fn:doc($uri as xs:string?) as document-node()?
```

Returns a document using an xs:anyURI supplied as xs:string. See the specification for details.

It is also possible to use a URI from another scheme, such as "http://website/path-to-document" or "file://path-to-document" if you have enabled external file access during construction of the `XmlManager` object.

dbxml:lookup*()

```
dbxml:lookup-metadata-index($containerName as xs:string, $metadataName as
xs:string) as document-node()*
```

The dbxml:lookup-* extension functions allow access to functions provided by the `XmlIndexLookup` class in the API. They can be used when very specific index lookups are required by a query, or the query plan is not sufficiently optimized for a given query.

The dbxml:lookup-index() extension function will lookup and return all elements with the name $elementName (and optionally parent $parentName) from the indices of the container named by $containerName. If appropriate indices do not exist, the empty sequence is returned. An error is raised if the specified container cannot be opened. This function can be used to perform presence index lookups directly from a query. Value, range, and substring index lookups can be used by adding predicates to the function call.

The dbxml:lookup-attribute-index() extension function will return all attributes with the name $attribute (and optionally the parent, $parentName) from the indices of the container named by $containerName. If appropriate indices do not exist, the empty sequence is returned. An error is raised if the specified container cannot be opened. This function can be used to perform presence index lookups for attributes directly from a query. Value, range, and substring index lookups can be used by adding predicates to the function call.

The dbxml:lookup-metadata-index() extension function will lookup and return the document nodes for documents that contain the metadata $metadataName from the indices of the container named by $containerName. If appropriate indices do not exist, the empty sequence

is returned. An error is raised if the specified container cannot be opened. This function can be used to perform presence index lookups for metadata directly from a query. Value, range, and substring index lookups can be used by adding predicates to the function call.

dbxml:metadata()

```
dbxml:metadata($qname as xs:string, $node as node()) as xs:anyAtomicType?
```

The dbxml:metadata() extension functions return the named metadata items from the document to which the specified node belongs. If the specified metdata does not exist in the document specified by $node, the empty sequence is returned, otherwise the type of the item returned is the same as the type of the metadata set in the document. If the context item is not a node, an error is raised [err:FODC0001].

dbxml:node*()

```
dbxml:node-to-handle($node as node()) as xs:string
```

dbxml:handle-to-node() will return the node from the specified container as referenced by the handle. If the specified container does not exist or cannot be opened an error is raised. An error is raised if the handle is invalid or the node specified does not exist.

dbxml:node-to-handle() will return an opaque node "handle" for the given node. The string returned is base-64 encoded raw data, and is suitable for embedding in a URI. The handle can be used to retrieve the node at a later time, if the node is still present in its container, using dbxml:handle-to-node().

Chapter 5. Upgrading Berkeley DB XML

This chapter describes how to upgrade from one version of BDB XML to another, when an upgrade is necessary.

Library Version Information

Each release of the Berkeley DB XML library has a major version number, a minor version number, and a patch number.

The major version number changes only when major portions of the Berkeley DB XML functionality have been changed. In this case, it may be necessary to significantly modify applications in order to upgrade them to use the new version of the library.

The minor version number changes when Berkeley DB XML interfaces have changed, and the new release is not entirely backward-compatible with previous releases. To upgrade applications to the new version, they must be recompiled and potentially, minor modifications made (for example, the order of arguments to a function might have changed).

The patch number changes on each release. If only the patch number has changed in a release, applications do not need to be recompiled, and they can be upgraded to the new version by installing the new version of a shared library or by relinking the application to the new version of a static library.

Internal Berkeley DB XML interfaces may change at any time and during any release, without warning. This means that the library must be entirely recompiled and reinstalled when upgrading to new releases of the library because there is no guarantee that modules from the current version of the library will interact correctly with modules from a previous release.

To retrieve the Berkeley DB XML version information, applications should use the `DbXml::dbxml_version` function. In addition to the previous information, the `DbXml::dbxml_version` function returns a string encapsulating the version information, suitable for display to a user.

Upgrading Berkeley DB XML Applications to 2.3

The database format has changed in Berkeley DB XML release 2.3 and existing containers created in 2.x-based applications must be upgraded, and applications should be recompiled.

Upgrading 2.x-based Containers

It is possible to upgrade containers created with 2.0, 2.1, or 2.2. It is not possible to upgrade containers created with a 1.x release of BDB XML. Containers can be upgraded in one of two ways. Using the `dbxml` shell facility, or using the `XmlManager::upgradeContainer()` method. The shell program method is simpler.

Once the installation is built, these steps can be used to perform the upgrade:

1. **Important:** Make a backup copy of your container.

2. Assuming the dbxml program is in your PATH, type `dbxml`.

3. **dbxml>** upgradeContainer <*pathToContainer*>

4. **dbxml>** quit

The upgrade may take a few minutes, depending on the size of your container. You can check to see if the upgrade was successful by opening the container with the
openContainer
sub-command in dbxml.

Depending on the contents, node storage containers may bloat during an upgrade process, resulting in a noticeably larger container. If this is undesirable, it is possible to shrink the size by performing a `dbxml_dump` and `dbxml_load` combination on the container, while is it not being accessed by any other programs.

Upgrading Berkeley DB XML Applications to 2.2

The database format has changed in Berkeley DB XML release 2.2 and existing containers created in a 2.0- or 2.1-based application must be upgraded. 2.1-based applications do not need to be recompiled in order to work with Berkeley DB XML release 2.2.

Upgrading 2.0- or 2.1-based Containers

It is possible to upgrade containers created with either 2.0 or 2.1. It is not possible to upgrade containers created with a 1.x release of BDB XML. Containers can be upgraded in one of two ways. Using the `dbxml` shell facility, or using the `XmlManager::upgradeContainer()` method. The shell program method is simpler.

Once the installation is built, these steps can be used to perform the upgrade:

1. **Important:** Make a backup copy of your container.

2. Assuming the dbxml program is in your PATH, type `dbxml`.

3. **dbxml>** upgradeContainer <*pathToContainer*>

4. **dbxml>** quit

The upgrade may take a few minutes, depending on the size of your container. You can check to see if the upgrade was successful by opening the container with the `openContainer` sub-command in dbxml.

Change in dbxml shell default container type

In 2.2 the default container type created by the `dbxml` shell program is `XmlContainer::NodeContainer`, with nodes indexed. Previously, the default was `XmlContainer::WholedocContainer`. There is no effect on existing containers. The only possible effect is on a script that depends on the default container type created, in which case the script should be changed to use an explicit type of "d." E.g.

dbxml> createContainer <*pathToContainer*> d

Upgrading Berkeley DB XML Applications to 2.1

The database format has changed in Berkeley DB XML release 2.1 and existing containers created in a 2.0-based application must be upgraded. In addition, applications must be recompiled in order to work with Berkeley DB XML release 2.1.

Upgrading existing containers

Existing containers can be upgraded in one of two ways. Using the `dbxml` shell facility, or using the `XmlManager::upgradeContainer()` method. The shell program method is simpler.

Once the installation is built, these steps can be used to perform the upgrade:

1. Make a backup copy of your container.
2. Assuming the dbxml program is in your PATH, type `dbxml`.
3. **dbxml>** upgradeContainer <*pathToContainer*>
4. **dbxml>** quit

The upgrade may take a few minutes, depending on the size of your container. You can check to see if the upgrade was successful by opening the container with the `openContainer` sub-command in dbxml.

Upgrading Berkeley DB XML 1.2.X applications to Berkeley DB XML 2.0

If you are new to Berkeley DB XML or are already using a 2.x release, this documentation on 2.0 migration may be skipped. Berkeley DB XML Release 2.0 represents a significant upgrade from earlier Berkeley DB XML releases. This document discusses the changes, and how applications written for earlier releases can be upgraded to 2.0. There are sections for both C++ and Java. The Java section only discusses a few Java-specific issues. Java users should read the C++ section as well. It is assumed that the reader is familiar with the 1.2.X release of Berkeley DB XML. While reading this section, it will be helpful to refer to the C++ API reference or the Javadoc.

Features that are new in Berkeley DB XML 2.0 are not discussed, except as they affect code changes between 1.2.X and 2.0. This document is not intended to be an introduction to Berkeley DB XML Release 2.0. For a complete introduction to BDB XML, see either the C++ or Java version of the Berkeley DB XML Getting Started Guide. For a complete description of the BDB XML API, see either the C++ API reference or the Javadoc.

New and Changed Features in 2.0

The number of new features added in Berkeley DB XML Release 2.0 has led to a new API, as well as new database format. These require application changes in order to take advantage of the features of 2.0. The major features and changes visible to an application include:

- XQuery support
- Introduction of an `XmlManager` object to manage XmlContainers, and act as a factory object for operation contexts, including queries
- Addition of a node storage container type
- `XmlDocument` metadata handling improvements
- Flexible interfaces for getting XML documents in and out of Berkeley DB XML
- Extended indexing, including additional types, metadata indices, and unique indices.
- General API improvements

A good way to become familiar with the new interface is to examine some of the 2.0 example programs. The next sections of this document use before/after code comparisons for common Berkeley DB XML operations to demonstrate how a 1.2.X program must change to work with 2.0.

Migrating Berkeley DB XML C++ Applications

XmlManager

The `XmlManager` is a new object in 2.0. It is used as a factory object for many Berkeley DB XML objects, as well as providing context for operations such as queries. Some of the common operations on `XmlManager` are:

- `XmlManager::createContainer()`
- `XmlManager::openContainer()`
- `XmlManager::createTransaction()`
- `XmlManager::query()`

Many of the operations that were previously methods on `XmlContainer` are now methods on XmlManager.

XmlContainer Management

The following is a comparison of 1.2.1 and 2.0 code to create an `XmlContainer` and insert a new document:

```
// Create a container, insert a document
// Do not use environment or transactions
//
// In 1.2.1
    XmlContainer container(0, "test.dbxml");
    container.open(0, DB_CREATE|DB_EXCL, 0);
    XmlDocument doc;
    doc.setContent("<root>newdoc</root>");
    container.putDocument(0, doc, 0);
```

```
    container.close(0);
//
// In 2.0
    XmlManager mgr;
    XmlContainer container = mgr.createContainer("test.dbxml");
    // createContainer and openContainer return opened containers
    XmlUpdateContext uc = mgr.createUpdateContext();
    container.putDocument("doc1", "<root>newdoc</root>", uc);
    // container and manager are closed when objects go out of scope
```

The points to notice are:

- `XmlManager` is a factory object
- 2.0 requires an `XmlUpdateContext` object for all modifications. This was happening under the covers in 1.2.X.
- 2.0 does not require creation of an `XmlDocument` object in order to insert content. It is still an option.
- 2.0 no longer exposes numeric document IDs. It requires names for documents, and the names must be unique within a container. The flag, DBXML_GEN_NAME, can be used to tell the system to generate a unique name if names are not important to an application.
- Valid `XmlContainer` objects are implicitly opened when created.
- Object scoping is used for automatic cleanup and exception safety. Internally, the 2.0 `XmlManager` and XmlContainer objects are reference counted, and closed upon release of last reference.
- `XmlManager` has an openContainer() method that must be used to open existing containers. It can also be used to create new containers.

XmlManager and Berkeley DB DbEnv

In the 1.2.X API, the `XmlContainer` constructor takes a DbEnv * parameter which is used if a DbEnv is required. In 2.0, the DbEnv (Berkeley DB environment) is associated with the `XmlManager` object. In 1.2.X, the DbEnv, if provided, is managed externally to Berkeley DB XML. In 2.0, there is an option of passing the flag, DBXML_ADOPT_DBENV. If the DbEnv is adopted, it is owned by the `XmlManager` object, and is closed when the `XmlManager` destructor runs:

```
DbEnv *env = new DbEnv(0);
env->open("path", DB_INIT_MPOOL|DB_CREATE, 0);
XmlManager mgr(env, DBXML_ADOPT_DBENV);
// XmlManager will close and delete the DbEnv
// object when it goes out of scope
```

Queries

The addition of the `XmlManager` object and the introduction of the XQuery query language to 2.0 change the way that queries are performed in two ways:

1. Query language is XQuery, and no longer XPath 1.0. Most XPath 1.0 queries are valid in XQuery, usually with the addition of additional required syntax in XQuery.

2. The Query context for 2.0 is the XmlManager object, and not constrained to a specific XmlContainer. A single XQuery can reference more than one container, and even reference specific documents, by name.

The following code compares a simple query in 1.2.X and 2.0:

```
// Assume an open container and XmlManager
// Assume container name is "test.dbxml"
// Do not use environment or transactions
//
// In 1.2.1
       XmlResults results(container.queryWithXPath(0, "/vendor", 0));
       XmlValue value;
       while (results.next(value)) {
           // do something
       }
//
// In 2.0
       // XmlQueryContext is required
       XmlQueryContext qc = mgr.createQueryContext();
       XmlResults results =
            mgr.query("collection('test.dbxml')/vendor", qc);
       XmlValue value;
       while (results.next(value)) {
           // do something
       }
```

The points to notice are:

- `XmlQueryContext` is required in 2.0. In 1.2.X, it was created under the covers if it was defaulted.
- The 2.0 query requires the string "collection('test.dbxml')" to point to a specific container. There are a number of ways to control query context in 2.0, both in the XQuery expression itself, and through the query interfaces.

Transactions

2.0 introduces a new object, XmlTransaction, which is used to wrap the Berkeley DB DbTxn object, and aids in internal transaction management. Rather than using an optional DbTxn * argument to a single interface, 2.0 defines 2 separate interfaces for each operation that may be transacted. One takes an `XmlTransaction &` argument, and the other does not. The following code compares 1.2.X and 2.0 code that performs a simple, transacted operation:

```
// Create a container, insert a document
// Use environment and transactions
// Assume DbEnv* has been constructed as dbEnv;
//
// In 1.2.1
```

```
        DbEnv *dbEnv;
        ...
        XmlContainer container(dbEnv, "test.dbxml");
        DbTxn *txn;
        dbEnv->txn_begin(0, &txn, 0);
        container.open(txn, DB_CREATE|DB_EXCL, 0);
        txn->commit(0);
        // new transaction for insert
        DbTxn *txn1;
        dbEnv->txn_begin(0, &txn1, 0);
        XmlDocument doc;
        doc.setContent("<root>newdoc</root>");
        container.putDocument(txn1, doc, 0);
        txn1->commit(0);
        ...
        container.close(0);
        dbEnv->close(0);
//
// In 2.0
        DbEnv *dbEnv;
        ...
        XmlManager mgr(dbEnv, DBXML_ADOPT_DBENV); // adopt env
        // create a transacted container
        XmlContainer container =
               mgr.createContainer("test.dbxml", DBXML_TRANSACTIONAL);
        XmlTransaction txn = mgr.createTransaction();
        // createContainer and openContainer return opened containers
        XmlUpdateContext uc = mgr.createUpdateContext();
        container.putDocument(txn, "doc1", "<root>newdoc</root>", uc);
        txn.commit();
```

The points to notice are:

- 2.0 adds a DBXML_TRANSACTIONAL flag that can be passed to createContainer() and openContainer() to avoid the necessity of creating and committing a transaction for this purpose.
- The DBXML_ADOPT_DBENV flag simplifies cleanup in 2.0.
- The use of `XmlTransaction and XmlManager::createTransaction()` allows an application to ignore DB objects for most operations.
- There is an `XmlManager::createTransaction()` method that takes a DbTxn * argument, allowing a DbTxn to be wrapped.

Migrating Berkeley DB XML Java Applications

The Java interface to Berkeley DB XML is similar to the C++ interface in spirit. In addition to the changes to the interface due to functional changes in Berkeley DB XML, the Java interface has changed to be more compatible with the Berkeley DB Java interface.

XmlManager and Environment

The Berkeley DB Java interface replaces the DbEnv object with an Environment object. It also replaces the DbTxn object with Transaction. The interface also replaces the use of integer flags with configuration objects. Berkeley DB XML has adopted this mechanism as well.

Configuration Object

There are 3 new configuration objects in Berkeley DB XML, replacing corresponding use of flags:

1. `XmlManagerConfig`

   Use this object to configure a new `XmlManager` object.

2. `XmlContainerConfig`

   Use this object to configure `XmlContainer` objects. A default `XmlContainerConfig` object can be set on an `XmlManager` object that affects all containers it creates and opens. This object extends `DatabaseConfig`, and inherits state, such as encryption, read isolation level, threading configuration, and read-only.

3. `XmlDocumentConfig`

   Use this object to configure XmlDocument-level state, such as `DBXML_LAZY_DOCS` or `DBXML_GEN_NAME` (C++ flags).

Delete, GC and Object Life Cycle

Because Java objects in Berkeley DB XML are wrappers for native (C++) objects, the Java VM is not aware of memory consumed by the native objects. Therefore, GC on objects may not happen in a timely manner, if at all. This can result in out of memory conditions, outside of the Java VM control.

For this reason, it is necessary to explicitly delete most Berkeley DB XML Java objects when they are not longer required. This is especially true for the XmlResult, XmlValue, and `XmlDocument` objects, of which there can be many. All of the objects include delete() methods for this purpose.

Migrating Berkeley DB XML Data to 2.0

The database format is new in Berkeley DB XML release 2.0, and there is no upgrade utility at this time. If it is not possible to reload data from external files, it is possible to write a small, custom application to dump 1.2.X data, and load it into 2.0. The Berkeley DB XML dbxml_dump and dbxml_load programs will not work for this purpose.

Information Necessary for Load into 2.0

Migrating data is best thought about in terms of what information is needed to load into 2.0. A load comprises the following operations:

1. Create a container. Choose a name, and type of container (node storage vs whole document storage — a new feature in 2.0).

2. Specify indices. The same indices from a 1.2.X container will work; however, 2.0 introduces a number of new options and index types that can be used.

3. Load XML documents. 2.0 requires names, and 1.2.X XmlDocument objects have numeric IDs, not names. The numeric IDs can serve as names, or the system can generate unique names, using the DBXML_GEN_NAME flag.

4. Load XML document metadata. 2.0 has changed, and extended handling of metadata, including metadata indices. Also, in 2.0, metadata is no longer part of the document.

Information to Dump from 1.2.X

The remaining task is thinking about how to dump a 1.2.X container such that the information above is available:

1. The `XmlContainer` name and type are an application choice, based on expected usage. If the application performed well with 1.2.X, then using whole document storage may be preferred.

2. `XmlIndexSpecification` information can be extracted from a 1.2.X container.

3. XML documents can be dumped to local files, for reloading.

4. Obtaining 1.2.X metadata is more difficult. In this case, the application needs to know that the metadata exists, and acquire it and dump it to a format that can be used in the load step.

Appendix A. Berkeley DB XML Changelogs

This appendix contains the Berkeley DB XML changelogs for both the current release and all historical releases.

Berkeley DB XML 6.0.18 Change Log

Release 6.0 is primarily a bug fix and infrastructure enhancement release. Including:

1. Reliability:

   Fixed many bugs, and enhanced product stability.

2. Indexing:

   The indexing module was reconstructed with a clear and high performance architecture.

3. Dynamic resource management:

   The release bundles Berkeley DB 6.1, which provides dynamic resource management. This is a big win for BDB XML users who tend to not want to manage DB resources.

Berkeley DB XML 6.0.18 Change Log

BDB XML 6.0.18 is a bug-fix release that addresses a number of issues found since the 6.0.17 release (see Berkeley DB XML 6.0.17 Change Log (page 45) for details). It is source and binary compatible with earlier 6.0.x releases. This section describes changes in BDB XML relative to release 6.0.17.

Upgrade Requirements

None relative to 6.0.17. See Berkeley DB XML 6.0.17 Change Log (page 45) for upgrade details and recommendations.

New Features

1. Added support for Java 8.[#24297]
2. Added support for Berkeley DB 6.1.26.[#24402]
3. Added support for XQilla 2.3.2.[#24402]
4. Added support for Xerces-C 3.1.2.[#24402]

API Changes

1. None.

Changes That May Require Application Modification

1. None.

General Functionality Changes

1. Fixed a bug that could result in variables in a query to be read as undefined when they are defined.[#24044]

Utility Changes

1. None.

Java-specific Functionality Changes

1. Added support for Java 8.[#24297]

Python-specific Functionality Changes

1. None.

Perl-specific Functionality Changes

1. None.

PHP-specific Functionality Changes

1. None

Example Code Changes

1. None.

Configuration, Documentation, Portability and Build Changes

1. Fixed dependencies in Visual Studio 10 so Xerces-C will automatically be built before XQilla. [#24297]

2. Fixed a build issue with Visual Studio 2013. [#24459]

Berkeley DB XML 6.0.17 Change Log

Upgrade Requirements

Upgrade is required for containers created using release 2.5.16 or earlier.

Because 6.0 bundles Berkeley DB 6.1, environment directories and log files will not be compatible with previous releases of Berkeley DB XML. This means checkpoint, backup and recovery procedures are necessary to start with a clean environment, especially for transactional environments.

If an upgrade is performed it is recommended that the resulting container be run through dbxml_dump/dbxml_load to reduce its file size.

New Features

1. The indexing module was reconstructed with a clear and high performance architecture. [#17397]
2. The release bundles Berkeley DB 6.1, which implements dynamic resource management. This is a big win for BDB XML users who tend to not want to know about DB resource manager.[#20161]
3. Add BDB XML support for DBXML_IGNORE_LEASE. This feature is not supported in Java API yet. [#19025]
4. Improved Unicode module and implement full Unicode-standard code-point support. [#17394]

API Changes

1. XmlIndexSpecification:

   Changed the behavior of bool `XmlIndexSpecification::next(std::string &uri, std::string &name, std::string &)` so that only one index string will be returned at each iteration.[#17397]

2. XmlManager:

   createTransaction() — Add DBXML_IGNORE_LEASE to support the Berkeley DB HA MasterLease feature.

3. XmlTransaction:

   createChild() — Add DBXML_IGNORE_LEASE to support the Berkeley DB HA MasterLease feature.

Changes That May Require Application Modification

1. BDB XML adds supports for Visual Studio 2010 (BDBXML_all_vs2010.sln). [#18693]
2. BDB XML no longer supports Visual Studio 6 and Visual Studio 2003, and changes the solution name of Visual Studio 2005/2008 from BDBXML_all_vs8.sln to BDBXML_all.sln. [#19003]
3. BDB XML no longer supports the IA64 platform. [#21089]

General Functionality Changes

1. This release bundles Berkeley DB 6.1. [#22506]
2. Fixed a bug that could cause a segfault for certain queries. [#22853]
3. Fixed a bug that would cause unique constraints to be ignored when updating part of an XML document. [#21239]
4. Do not created nested transactions when updating documents. [#18220]

5. Fixed a possible mutex lock leak when closing a container.[#18959]
6. Handle error conditions that may occur when replication is enabled [#18960]
7. Fixed a possible case of deadlock when opening a container while another process was inserting documents into that container.[#19009]
8. Fixed a bug where an external function that constructed documents would return empty results [#18772]
9. Adding new linear sort algorithm for BulkPut class. [#17417]
10. Fixed an issue that may cause a crash in a limited memory system.[#18108]
11. Added a "-lm" flag when link on Solaris with SunCC 5.9 [#18171]
12. Fix the issue that prevented upgrading 2.3 containers.[#18741]
13. Fix some issues with external functions where they were not getting the proper cache database for intermediate results.[#18772]
14. Fix a bug in `DbXmlNamespaceAxis::nextNode()` that caused the namespace uri to be returned more than once for the prefix "xml".[#18996]
15. Fixed a bug where mixing auto-indexing and default indexes could result in incorrect query results.[#19048]
16. Delete operations now update stats.[#19102]
17. Fixed a segfault when calling doc-available() in XQuery function.[#19284]
18. Fixed a segfault when an XML document is blank.[#19398]
19. Fixed a bug which returns unexpected duplicated results when querying parents or ancestors of an element or attribute node.[#19717]
20. Fixed a query optimization bug that could happen when comparing attribute children of an element.[#20057]
21. Fixed rounding issue when converting double to int.[#20066]
22. Fixed an incorrect flowr expression with `DbXml::XmlContainer::WholedocContainer`.[#19563]
23. Fixed a bug involving the copying of dummy MetaData items. [#20126]
24. Fixed a bug where it was not possible to reference metadata with an `XmlQueryExpression` executed against an `XmlDocument` object. [#20987]
25. Optimize the query optimizer for mixing attribute and elements in a where clause. [#20830]
26. Fixed a crash that can occur when using DB_READ_COMMITTED or DB_READ_UNCOMMITTED. [#21601]

27. Fixed a bug were inserting a node could erase attribute index entries of the parent node. [#21772]

Utility Changes

1. None.

Java-specific Functionality Changes

1. The Java API supports Java 7 and Java 8. [#22517]

Python-specific Functionality Changes

1. The latest Python code from the pybsddb project is bundled.

Perl-specific Functionality Changes

1. Upgraded Perl version to 5.16 for the Windows binary release. [#20495]

PHP-specific Functionality Changes

1. Removed PHP from the Windows binary release. [#21548]

Example Code Changes

1. None.

Configuration, Documentation, Portability and Build Changes

1. Removed support for Visual Studio 2003. [#19003]
2. Removed support for IA64. [#21089]
3. BDB XML bumps the bundled Berkeley DB version from 4.8 to 6.1. Berkeley DB 6.1 includes a large number of behavioral changes. Please refer to the Berkeley DB changelog to identify the new HA features offered by Berkeley DB 6.1.

2.5 Release Overview

Release 2.5 is primarily a feature release with a small number of useful features including:

- Automatic indexing of leaf elements and attributes
- Whole Document container compression with optional user-defined compression in C++ and Java
- Improvements in node storage containers that reduce total size of containers
- User-defined external XQuery extension functions in C++, Java and Python
- XQuery debug API in C++, Java and Python
- Improvements in the `XmlResults` class enabling better offline results handling

Berkeley DB XML 2.5.16 Change Log

BDB XML 2.5.16 is a bug-fix release that addresses a number of issues found since release of 2.5.13. It is source and binary compatible with earlier 2.5.x releases. This section describes changes in BDB XML relative to release 2.5.13.

Upgrade Requirements

None relative to 2.5.13. See 2.5.13 for upgrade details and recommendations.

General Functionality Changes

1. Upgraded the packaged version of Berkeley DB to 4.8.26. Please see the Berkeley DB specific change log for relevant changes.

2. Fixed container creation so that it honors page size in `XmlContainerConfig` [#17803]

3. Fix the base-uri of an attribute node when using WholedocContainer storage [#17872]

4. Fixed an assertion failure during query preparation with a recursive user defined function [#17866]

5. Fixed an assertion failure when an as-yet unseen URI is in use in a query [#17867]

6. Fixed a problem where attribute indexes would not properly be updated if there were no element indexes present. This might have a symptom of DB_NOTFOUND errors or duplicate index entries for attributes [#17671]

7. Changed the algorithm used to create node IDs during partial update to be more efficient and create shorter node IDs in general [#17844]

8. Fixed a problem where deleting the `XmlResults` object returned by XmlValue.getAttributes() might cause an exception when the original `XmlResults` for the `XmlValue` object was next accessed [#17796]

9. XQuery Update queries will no longer crash when statistics are disabled [#17898]

10. Fixed a bug in document level indexing that could result in index entries being deleted inappropriately when a node was deleted [#17758]

11. Fixed fn:doc() to raise an error in all cases if the document does not exist [#17870]

12. Fixed a bug occurring when fn:subsequence() and "order by" were used in certain configurations [#17932]

13. Changed XmlResults.asEventWriter(), now only one active `XmlEventWriter` is allowed for an `XmlResults` object [#18049]

Utility Changes

None.

Java-specific Functionality Changes

1. Deleting the `XmlResults` returned by `XmlValue.getAttributes()` will no longer cause exceptions to be thrown when accessing other XmlResults.[#17792].

2. Fixed a few memory leak issues in JNI code. Fixed leaks may happen in XmlEventWriter.writeXXX() methods, XmlDocuments.getContent(), XmlEventReaderToWriter.start(), XmlResults.copyResults(), XmlResults.concatResults(). [#18049].

Python-specific Functionality Changes

1. The latest Python code from the pybsddb project is bundled.

Perl-specific Functionality Changes

None.

PHP-specific Functionality Changes

None.

Example Code Changes

None.

Configuration, Documentation, Portability and Build Changes

1. The zlib library will now be copied to both the release and debug directories on Windows. [#17894]

Berkeley DB XML 2.5.13 Change Log

Upgrade Requirements

Containers do not require upgrade; however because 2.5 bundles Berkeley DB 4.8 environment directories and log files will not be compatible with previous releases. This means checkpoint, backup and recovery procedures are necessary to start with a clean environment especially for transactional environments.

New Features

1. Automatic indexing of leaf elements and attributes. When a container is set in the auto-indexing state it will automatically detect new leaf elements and attributes and add string and double indexes for them (node-*-equality-string and node-*-equality-double). This feature has been added to enhance out-of-the-box performance of queries and can be used to replace default indexes in most cases. Default value indexes have a tendency to over-index mixed content. Newly-created containers will be in this state, which is controlled by new API outlined below; some applications may wish to disable automatic indexing immediately after creating a container. Addition of new indexes can be a significant operation so this behavior is best for containers that store similar documents or it must be carefully controlled by the application. When enabled, addition of any

new content can trigger the equivalent of `XmlContainer::setIndexSpecification()`. [#15722]

2. Data compression for whole document storage. XML documents can now be compressed using a built-in implementation based on the zlib compression library or user defined compression created by implementing the class `XmlCompression`. [#15471]

3. Debugging API and command line debugger. The command line debugger can be started using the "debug" command in the DB XML shell. The debugging API is used by deriving a class from `XmlDebugListener` and registering it with the `XmlQueryContext::setDebugListener()` method. Access to the stack trace and dynamic context in each stack frame is available through the `XmlStackFrame` class during debugging. [#15999]

4. XQuery external functions. The `XmlExternalFunction` class allows the implementation of external XQuery functions in C++, Java and Python. Users should derive a class from XmlExternalFunction, implementing the execute() method to perform the function's action. Arguments are provided to the execute() method via the `XmlArguments` class. [#15610]

5. Enhanced results handling

   - Added `XmlResults::asEventWriter()` which can be called on an empty (newly-created) `XmlResults` object to construct a sequence of elements and/or atomic values that can be used in queries. The constructed object must be assigned to a variable to be used. [#16355]

   - Added `XmlResults::copyResults()` and `XmlResults::concatResults()` which can be used to create and use "transient" copies of `XmlResults` which can be used outside of the context of a transaction or container.

6. Better node storage efficiency. The storage algorithm for node storage containers has been modified to provide a better btree fill factor, resulting in smaller container files

API Changes

Unless otherwise noted, the API additions apply to all language bindings, and all bindings use the same method name.

1. There are new classes for the external XQuery function implementation noted above under features. These are `XmlExternalFunction` and `XmlArguments` and are available only in C++, Java and Python.

2. New interfaces control automatic indexing behavior:

   The modified `XmlIndexSpecficiation` instance must be set on the container using `XmlContainer::setIndexSpecification()` in order for the state change to occur. The state is persistent and is set to true for newly-created containers.

   - `XmlIndexSpecification::setAutoIndexing(bool)` and `XmlContainer::setAutoIndexing()` Use these to control automatic

indexing. The method on `XmlIndexSpecification` will only take effect once `XmlContainer::setIndexSpecification()` is called with the modified `XmlIndexSpecification` object. The method on `XmlContainer` is a convenience method that uses its underlying `XmlIndexSpecification`.

- `bool XmlIndexSpecification::getAutoIndexing()` and `XmlContainer::getAutoIndexing()` Returns the current state of automatic indexing for the container. The method on XmlContainer is a convenience method that returns the state from the XmlIndexSpecification

3. Added `XmlContainerConfig` to the API. XmlContainerConfig already existed in the Java API but has been expanded and is now available in all other language bindings. `XmlContainerConfig` simplifies opening and creating of containers and is intended to replace the flags arguments to operations that create and open containers. The flags versions still exist for now but will eventually disappear.

4. Added `XmlCompression` as a class that can be used to create user-defined compression algorithms for wholedoc containers. This class is available only in C++ and Java

5. Added constructor `XmlValue(typeURI, typeName, value)` for creating atomic values with derived types.

6. Added new functions to `XmlData` and changed it so that it functions as an actual buffer for binary data rather than a wrapper for an existing buffer.

7. Added the functions `XmlValue.getResults()` and `XmlDocument.getResults()` to the Java API to return the `XmlResults` object (if any) associated with the current object [#16352]

8. Removed all `finalize()` functions from the Java API. They served no useful purpose and could only cause problems by running at inconvenient times [#16352]

9. Added `XmlResults::asEventWriter()` — see description above under "Features" [#16355]

10. Configuring a database as an XA-compliant resource manager using the flag DB_XA_CREATE is no longer supported because XA support has been removed from Berkeley DB [#16912]

11. `XmlModify` has been removed, XQuery Update should be used instead [#16915]

12. The Berkeley DB C++ objects in the C++ API have been replaced with their C equivalents. This simplifies infrastructure and improves build/linking with the non-C++ interfaces. It requires changes for C++ applications where they may have used DbEnv or DbTxn [#16951]

13. Added the function `XmlIndexSpecification.getValueType(index)` that returns the `XmlValue::Type` described in the given index description. [#17362]

Changes That May Require Application Modification

1. C++ applications are required to change the use of the Berkeley DB C++ objects in the public interface to their C equivalents. Such changes are mechanical, replacing

DbEnv* with DB_ENV * and DbTxn * with DB_TXN * and only a few interfaces are affected. For example, XmlManager::XmlManager(DbEnv *, u_int32_t) becomes XmlManager::XmlManager::(DB_ENV *, u_int32_t). The DbEnv object has a method, DB_ENV *DbEnv::get_DB_ENV() that can be used. Similarly, DbTxn has a method DB_TXN *DbTxn::get_DB_TXN() that can be used.

2. Automatic indexing may require changes. It is enabled on newly-created containers by default and if an application wishes to not have this feature enabled it will need to explicitly disable it, post-creation. Most applications will eventually want to disable this state once they are confident that all useful indexes have been added. An application that wants very explicit control over its indexes should disable it. If it is not desired at all then immediately after creating a container call execute this sequence of operations (some pieces are missing and this example does not use transactions but these are the calls):

```
XmlIndexSpecification is = container.getIndexSpecification();
is.setAutoIndexing(false);
container.setIndexSpecification(is, updateContext);
```

3. If compression is compiled in, which is the default, wholedoc containers are by default compressed using zlib compression. This can be disabled using interfaces on `XmlContainerConfig` when creating the container. Existing containers are not affected

4. While the addition of `XmlContainerConfig` to the non-Java APIs does not require change it is recommended that applications move to the methods that use `XmlContainerConfig` as eventually the old interfaces will be phased out

5. The functions in the `XmlData` class that use the Dbt object have been removed, including one constructor, `set_data` and `getDbt`

6. Java objects that require clean up must be cleaned up manually by calling the `delete()` function. Failure to clean up objects can result in memory leaks and the need to run database recovery. This has always been the case but removal of finalizers has made it even more important for memory leak situations

7. The functions `XmlContainer.addIndex` and `XmlIndexSpecification.addIndex` will now throw an exception if passed the index types `XmlValue.DAY_TIME_DURATION`, `XmlValue.YEAR_MONTH_DURATION`, or `XmlValue.UNTYPED_ATOMIC`. If indexing the types `XmlValue.DAY_TIME_DURATION` or `XmlValue.YEAR_MONTH_DURATION` use `XmlValue.DURATION`. If indexing the type `XmlValue.UNTYPED_ATOMIC` use `XmlValue.STRING`. [#17365]

General Functionality Changes

1. The release bundles Berkeley DB 4.7.25

2. Added a "--disable-rpath" option to the configure script, to facilitate building embedding rpath information in libraries [#16607]

3. Fixed an uninitialized variable in NsEventWriter that could affect use of `XmlEventWriter` [#16459]

4. Fixed a bug where putting a document from one container into another would result in an empty document in the second container [#16456]

5. Fixed a problem where a deadlock exception in `XmlEventWriter` would mistakenly be reported as EINVAL and lost [#16343]

6. Fixed a bug where inserting a new root element into a document would not properly index the new content [#16500]

7. The behavior of XQuery Update (and `XmlModify`) was changed so that multiple document elements are no longer allowed. XQuery Update can also no longer be used to remove the document element to create an empty document. Such documents can still be created but only via `XmlContainer::putDocument()` and `XmlContainer::updateDocument()` [#16500]

8. Fixed a problem where text (comment, PI, text) updates that affect elements that own multiple text nodes could trigger an assertion failure or bad memory reference [#16543]

9. Fixed a problem where the behavior of eager and lazy results iteration was not consistent [#16484]

10. Fixed a bug where constructed documents could not be created from an XmlInputStream. [#16593]

11. `XmlValues` created from an empty document will no longer crash on calls to certain functions. [#16608]

12. `XmlInputStream` will no longer lose its source if the `XmlDocument` it came from is deleted. Also, `XmlDocument.getContentAsXmlInputStream()` will now always consume the content of constructed documents. [#16617]

13. Fixed a static initialization problem that appears on some Windows platforms related to NsNid and results in an exception during `XmlManager` construction. [#16565]

14. Fixed an assertion triggered when using a predicate against a variable containing constructed nodes. [#16556]

15. Fixed a problem where variable references to deleted nodes could lead to problems or incorrect behavior [#16583]

16. Fixed a segmentation fault that could occur if the last step of a path was a comparison or contains() function. [#16772]

17. The flags DBXML_ENCRYPT and DBXML_CHKSUM will no longer result in an exception when used correctly. [#16677]

18. Fixed an assertion failure that could happen when using numeric predicates. [#16775]

19. Fixed a problem where a transactional XQuery Update expression using fn:put() would fail during the transaction commit, indicating that the transaction was already committed[#16808]

20. Change the close() methods in `XmlEventReader` and XmlEventWriter to be pure virtual. Implementors of XmlEvent* must implement the close() method which may need to delete "this" in order to free the memory. [#16771]

21. Fixed `XmlContainer::putDocument()` and updateDocument() on wholedoc containers to ensure that new namespace uri prefixes are added to the dictionary. This could result in an exception during queries of read-only content or stray updates during read operations [#17212]

22. Fixed a problem in partial updates where a delete of a node when it has an ancestor with a presence index could result in removal of the ancestor's index, resulting in incorrect query results [#17199]

23. Fixed some issues in partial updates that might result in problems with indexes or missing records in the case where mixed content was being indexed and a descendent node is deleted or modified [#17226]

24. Improved performance of partial reindexing when inserting a new element into a node that already has a large number of child elements [#17393]

25. The functions `XmlContainer.addIndex` and `XmlIndexSpecification.addIndex` will now throw an exception if passed the index types `XmlValue.DAY_TIME_DURATION`, `XmlValue.YEAR_MONTH_DURATION`, or `XmlValue.UNTYPED_ATOMIC`.

26. Fixed problems with partial updates and statistics that affected both partial update performance and query plans resulting in lowered performance after a number of updates[#17393]

27. Fixed `XmlManager::compactContainer()` so that space made available is released to the file system on platforms that support this behavior [#17658]

28. Fixed an optimizer issue where certain range queries might not use an index if appropriate [#17649]

29. Fixed a partial update scenario where indexes could get corrupted resulting in DB_NOTFOUND errors during queries or index lookups. This could only occur when inserting multiple elements into the same parent node and not all the time [#17649]

Utility Changes

1. dbxml shell subcommands that reflected the `XmlModify` interface have been removed

Java-specific Functionality Changes

1. Fixed a bug where accessing `XmlValue` objects created from XQuery constructed nodes would cause a crash. [#16403]

2. Added the functions `XmlValue.getResults()` and `XmlDocument.getResults()` to the Java API. [#16352]

3. Fix a bug where updating queries that use nodes as variables would cause the JVM to crash. [#16583]

4. Fix a bug where setVariableValue in Java API use `XmlResult::size()` which lazily evaluated does not support.

5. Eliminated the possibility of `XmlResolver` objects being garbage collected while the object is still needed.[#16595]

Python-specific Functionality Changes

1. The latest Python code from the pybsddb project is bundled.

2. Modified interfaces that can legitimately return a NULL value (in C++ or Java) to return None in Python. [#16678]

3. Fixed exception class constructors for `XmlDatabaseError` and `XmlException`. Arguments were out of order. [#16628]

4. Fixed a bug in the Python bindings for `XmlEventWriter::writeText()` [#16626]

5. Fixed a typo that made `XmlInvalidValue` exception unavailable[#16711]

Perl-specific Functionality Changes

1. The Perl Db module included with BDB XML still uses the Berkeley DB C++ API. This is not a change but a non-change that is the only place remaining in the product bundle that still uses the Berkeley DB C++ API. This does not affect other languages at all.

2. Added `--perl-installdir` to the buildall.sh script to allow users to change the installation directory for perl packages

PHP-specific Functionality Changes

None.

Example Code Changes

1. C++ code uses `XmlContainerConfig` rather than flags

2. C++ examples have been rewritten to use the Berkeley DB C API where appropriate

3. Examples have been added to illustrate external XQuery functions in C++, Java and Python

4. Examples have been added to illustrate use of Wholedoc container compression in C++ and Java

5. Examples have been added to illustrate use of the XQuery debug API in C++, Java and Python

6. Examples have been added to illustrate use of Berkeley DB XML with threads and in a server in Java

Configuration, Documentation, Portability and Build Changes

1. The build system for the BDB XML library on *nix now uses automake for better maintainability and portability

2. Fixed `XmlEventReader` documentation to properly indicate that empty elements will not result in an EndElement event [#17213]

2.4 Release Overview

The major focus areas for release 2.4 include:

- Implementation of declarative update expressions, conforming to the Last Call Working Draft of W3C's XQuery Update 1.0.
- Scalability improvements in terms of reducing memory footprint required during query processing and results handling.
- Performance improvements by using more intelligent, cost-based query processing.
- Performance improvements by using iterator-based processing instead of tree-processing.
- General bug fixes and improvements.

Changes in BDB XML 2.4.16

BDB XML 2.4.16 is a bug-fix release that addresses a number of issues found since release of 2.4.13. It is source and binary compatible with earlier 2.4.x releases. This section describes changes in BDB XML 2.4.16 relative to release 2.4.13.

Upgrade Requirements

None relative to 2.4.13. See 2.4.13 for upgrade details and recommendations.

General Functionality Changes

1. Fixed a problem in the BDB XML XQuery Extension function dbxml:node-to-handle() where it could return an exception/error when attempting to create a handle for an attribute node. [#16180]

2. Fixed a bug in query optimization which resulted in exponential query preparation time increases as the number of negative predicates used was increased. [#16181]

3. Fixed a problem where it was not possible to open a container using the DB_RDONLY flag. [#16186]

4. Fixed a problem related to [#16186] where it was not possible to open a replicated container because they are implicitly marked read-only. [#16394]

5. Fixed the generation of an incorrect query plan for a numeric predicate in a where clause, and a memory corruption when JIT optimizing a DecisionPointQP nested inside another DecisionPointQP.[#16206]

6. Fixed a query problem where expressions starting with "/" would succeed on XML without a document node (e.g. constructed XML) when they should fail. [#16230]

7. Fixed a bug where metadata indexes would not be used in query optimization. [#16273]

8. Fixed a bug in node construction across module boundaries. [#16288]

9. Fixed a bug where queries that performed multiple concurrent sequencial scans on a whole document container would return incorrect results. [#16438]

10. Fixed a bug in optimization of certain query predicates that resulted in an inappropriate re-ordering of the results. [#16287]

11. Added a parameter to the QueryPlan optimisation methods that decreases the number of optimisations returned with the depth of the query plan. This limits the exponential growth of optimisation time as queries get bigger. [#16321]

12. Fixed a problem in wholedoc containers iterating over the same document multiple times. [#16328]

13. Fixed a bug in the BufferQP optimisation logic.[#16330]

14. Fixed user defined default element namespaces in XmlQueryContext. [#16336]

15. Fixed querying of a no-content document to not SEGV. [#16338]

16. Fixed a bug where insertion of new content could result in a document that appeared fine but would no longer query correctly under all circumstances. [#16340]

17. Fixed a problem where valid URIs passed to fn:put() would appear invalid. [#16254]

18. Fixed a bug in optimization of certain predicates that resulted in an inappropriate document order re-ordering of the results. [#16287]

19. Fixed a bug in node construction accross module boundaries. [#16288]

20. Eliminated overly-aggressive flags checking in calls to `XmlManager::reindexContainer()`. This prevented certain types of reindexing, specifically from Java where the C++ flags are not under direct user control.

21. Allow DBXML_WELL_FORMED_ONLY flag for all `XmlContainer::putDocument()` variants.

22. Fixed preprocessor macros involving _MSC_VER that caused some platform/compiler combinations to fail to build. [#16354]

23. Fixed a SEGV that could result from an invalid XQuery Update expression [#16373]

24. Fixed a bug due to a typo in the optimisation code for UnionQP. [#16402]

25. Fixed a race condition when reading the DecisionPointQP linked list. [#16413]

26. Modifications to reduce memory consumption of prepared queries.

27. Fixed a bug in node size calculations that resulted in incorrect statistics in the structural statistics database, leading to gradual query performance degradation if lots of documents were removed or updated in place. [#16405]

28. Fixed a problem in which results obtained via `XmlContainer::getAllDocuments()` or other index lookup operation, when used as context for an `XmlModify` or XQuery Update operation could cause a self-deadlock/hang. This would only occur when using transactions. [#16397]

29. Added DB_RMW flag to transacted document updates and deletes to reduce potential for deadlock. [#16464]

30. Fixed and assertion failure (SEGV in release mode) related to using the count() function in a predicate in the where clause [#16470]

31. Allow DBXML_WELL_FORMED_ONLY flag in `XmlIndexLookup::execute()` and `XmlContainer::getAllDocuments()` [#16518]

32. Fixed a problem related to partial reindexing during XQuery Update that could result in DB_BUFFER_SMALL errors among others [#16551]

Java-specific Functionality Changes

1. Fixed problem in the `XmlQueryContext` copy constructor where the `XmlManager` object was not being copied [#16177]

2. Fixed a problem where an XmlValue.getAttributes() call on an `XmlValue` object that was not the result of a query (e.g. directly from an XmlDocument) could result in an error [#16179]

3. Fixed a problem where XmlManager.reindexContainer() might fail a flags check when it should not [#16221]

4. [#16236]. Fixed a Java-specific problem where some context queries against constructed XML could cause a SIGSEGV.

5. Fixed a problem that could cause a crash in the JVM if a null `XmlUpdateContext` object were passed.[#16222]

6. Fixed a problem where XmlDocument.setContent(String) could improperly store characters, resulting in an exception when attempting to insert the document into a container. [#16257]

7. Fixed a bug with `XmlDocument` that could result in the metadata no longer being accessible after the document content was retrieved. [#16296]

8. Made the `XmlDocument` copy constructor public. [#16353]

9. Fixed unnecessary materialization/parsing of wholedoc documents during `XmlResults` iteration [#16517]

PHP-specific Functionality Changes

1. Added missing createLocalFileInputStream() to `XmlManager` API. [#16351]

Perl-specific Functionality Changes

1. Fixed Perl tests to run correctly

Utility Changes

1. Changed dbxml shell to not allow opening or creating of containers in an explicit transaction. In a transactional shell those operations will implicitly open the container in a transaction. The additional transaction can only create problems. [#16408]

Berkeley DB XML 2.4.13 Change Log

Upgrade Requirements

Upgrade is only required for containers created using release 2.2.13 or earlier. Containers creating using 2.3.X do not require upgrade. However, most queries will benefit from reindexing 2.3.X-based containers to add new structural statistics information used in query cost analysis so it is highly recommended. Reindexing is `required` in order to enable the substring index to be used on 1- and 2-character strings (a new feature in 2.4).

Reindexing should generally be performed offline as it is an expensive operation that reads all content and regenerates index databases.

If an upgrade is performed (e.g. from 2.2.13) it is recommended that the resulting container be run through dbxml_dump/dbxml_load to reduce its file size.

New Features

1. Conformance to Last-Call Working Draft of XQuery Update 1.0
2. Added the ability to use "document projection" when querying whole-document containers. This performance and memory optimization results in only materializing that portion of the document relevant to the query.
3. Partial document modifications will now only reindex those portions of the document(s) affected by the modification itself. This is a significant performance enhancement for partial update of large documents.

API Changes

Unless otherwise noted, the API additions apply to all language bindings, and all bindings use the same method name.

1. Added the DBXML_DOCUMENT_PROJECTION flag to the various query interfaces to enable use of this feature. In Java, this behavior is controlled by XmlDocumentConfig.setDocumentProjection().
2. Added a new XQuery extension function, dbxml:contains(), that allows case- and diacritic-insensitive string searches and can be optimized by a substring index.

3. Removed all C++ interfaces that used Xerces-C DOM, including:

   - `XmlDocument::setContentAsDOM()`
   - `XmlDocument::getContentAsDOM()`
   - `XmlValue::asNode()`

4. XmlModify is now a deprecated (but still-supported) class. XQuery Update should be used instead. One method has been removed as it is no longer supported by the internal infrastructure: XmlModify::setNewEncoding()

5. Added a new constructor to `XmlEventReaderToWriter` that allows an `XmlEventWriter` instance to be multiple-use. This makes it possible to write to it from multiple reader sources to concatenate content, for example.

6. Removed XmlUpdateContext::{get,set}ApplyChangesToContainers() methods. This behavior is no longer controllable. Changes to documents that are in containers will always be written. If transient changes are required, content must be copied (XQuery Update has syntax to do this directly)

7. Removed unused variant of XmlValue::asString(std::string &encoding). This variant never actually changed the encoding [#15822]

8. Added DBXML_STATISTICS and DBXML_NO_STATISTICS flags to enable/disable creation of an additional statistics database that is used for query optimization. The default is to create the database. Upgraded containers will NOT have a statistics database added unless they are explicitly reindexed. The cost of this optimization is a bit of extra work during document insertion. In Java this behavior is controlled by XmlContainerConfig.setStatisticsEnabled().

9. Added the DBXML_NO_AUTO_COMMIT flag, which can be specified to the `XmlQueryExpression::execute()` methods to turn off auto-commit of update queries when it is not appropriate.

10. Some `XmlException` error codes have changed - DOM_PARSER_ERROR and NO_VARIABLE_BINDING have been removed, XPATH_PARSER_ERROR is now called QUERY_PARSER_ERROR, and XPATH_EVALUATION_ERROR is now called QUERY_EVALUATION_ERROR. [#15792]

11. The enumeration, XmlQueryContext::DeadValues, has been removed. The related method, XmlQueryContext::setReturnType() remains but is a no-op. All results are LiveValues. This will not affect the vast majority of applications.

12. Added `XmlQueryExpression::isUpdateExpression()` to allow users to know whether an expression is updating or not.

Changes That May Require Application Modification

1. Some of the API changes above may result in the need to make minor code changes

2. Not all modification patterns that use `XmlModify` will continue to work given the new infrastructure. Specifically, operations that would both copy and delete the same content

(emulating a "move" operation) may not work. In all cases, such code can be rewritten to use XQuery Update directly, resulting in simpler code. Special attention should be paid to multi-step operations that include such side effects.

3. Changed default indexing type on containers to be node indexes for node storage containers and document indexes for document storage containers [#15863].

4. Java only — the function XmlContainer.getNode() function has changed its signature and will require code change if used. See details below under "Java-specific Functionality Changes."

General Functionality Changes

1. Partial document modification will result in only reindexing those portions of document(s) affected by the modification

2. The system now keeps better cost information and statistics and the query optimizer uses this information to perform more effective cost-based optimization

3. The content processing internals have been reworked to make heavy use of iterators and temporary Berkeley DB databases to significantly reduce the memory footprint of query handling as well as reduce the number of objects created and destroyed by a query. This leads to a more scalable, high-performance system

4. Substring indexes will now work on any length search string (e.g. 2-char) rather than be restricted to a 3-character minimum. Reindexing the container is required to get this functionality.

5. Various fixes and memory leak elimination in XmlEventWriter[#15405]

6. Fixed a problem where removing a default index could remove index entries for an overlapping non-default index[#15412]

7. Changed semantics of `XmlQueryContext::setNamespace()` to treat an empty namespace prefix as the default element namespace[#15630]

8. Fixed problem in `XmlModify` that could result in malformed XML if a prefixed element name were used without a mapping for that prefix[#15586]

9. Fixed URI resolution code to
not
add the base URI when the URI being resolved is absolute[#15583]

10. Fixed code to force explicit transactions (vs auto-commit) when using `XmlContainer::putDocumentAsEventWriter`. This is necessary because of the 2-part nature of this interface [#15578]

11. Fixed a crash that could occur if XmlResults.next() were called at the end of a result set [#15621]

12. Fixed a bug where XQuery expressions involving unused global variables were not being optimized correctly [#15661]

13. Fixed problem in `XmlEventReader::nextTag()` where it would mistakenly throw an exception on character data. Also changed semantics of `XmlEventReader` to always return start and end document events so that callers can know when content starts and ends [#15686]

14. Fixed case where the '>' character was not being escaped properly (according to the XML specificiation). This case is when it occurs in the sequence, "]]>" [#15739]

15. Fixed a problem in `XmlModify` where removing a node that was the last child and had leading text could cause a SEGV [#15615]

16. Enhanced `XmlEventReaderToWriter` API to not unconditionally close the `XmlEventWriter` object, allowing a single `XmlEventWriter` to be used more than once via that API. This allows `XmlEventReaderToWriter` to be used for example, to coalesce a number of results into one document [#15446]

17. Fixed a problem with `XmlIndexLookup` where a GT lookup that happens to start with the last entry in the index might return results when it should return none[#15408]

18. Fixed a bug which incorrectly reported an error for fractional seconds when the seconds filed was "59" [#15389]

19. Fixed a problem in statistics calculation for substring indexes that could cause a crash in fn:contains() [#15823]

20. Fixed a bad exception that might be thrown when inserting a schema-invalid document, due to the length of the error message [#15824]

21. Fixed a problem where a query that uses an `XmlDocument` that has just been "put" into a continer as context for the query might hang if done in the same transaction as the putDocument call[#15905]

22. Fixed a problem where querying empty CDATA sections could cause an assertion failure or bad memory reference[#15906]

23. Fixed a latent bug that could result in missing index entries after updateDocument or modifyDocument call. This is very obscure and has never been seen by a user. It requires an odd combination of indexes and updates[#15943]

24. Fix open/close race condition on XmlContainer. An application that concurrently opened/closed `XmlContainer` objects (not recommended...) could possibly reference bad memory [#15890]

25. Fixed several memory leaks that could occur if deadlock exceptions are thrown during document processing (most likely put and delete)

26. All update operations now work inside internal child transactions to ensure that they are properly aborted if necessary. This is not user visible

27. Internal buffer size for DB get operations on nodes is tuned to the calling operation (bulk vs single get)[#15607]

28. Fixed a bug in `XmlEventWriter` where the behavior was dependent on an uninitialized variable [#15968]

29. Changed dbxml_load_container to take a '-e' flag that causes the program to stop document loading in the event of a parse error. The default is to continue with the next document [#15777].

30. Fixed problem in `XmlModify` where newly-inserted element content could cause a bad memory reference and/or crash while calculating a new node id [#15974].

Utility Changes

1. The dbxml shell added commands:

   - setProjection allows control of the document projection feature

2. The dbxml shell can be invoked using the #! syntax in a *nix shell command, e.g. with the first ilne: #!<absolute_path>/dbxml -s

3. Handling of '#' comment lines in the dbxml shell has been improved so that they can occur anywhere in a line [#15689]

Java-specific Functionality Changes

1. Fixed a problem where committed or aborting a transaction that was already committed or aborted could crash, especially after a failed XmlManager.openContainer() call [#15729]

2. Is it no longer be necessary to explicitly delete objects of type `XmlValue`, `XmlDocument`, `XmlQueryContext`, `XmlMetaData`, `XmlMetaDataIterator` and `XmlUpdateContext`. They are implemented entirely as pure Java objects with no native memory to release. It will still be necessary to explicitly delete other Java objects to release native memory. In general the validity of `XmlValue` and `XmlDocument` objects returned via `XmlResults` (queries, index lookups, etc) is under control of the `XmlResults` object. When the `XmlResults` object is deleted node values that may have been associated with that object may no longer be accessible and an exception will be thrown if accessed [#15194]

3. Added -source 1.5 -target 1.5 to Java builds to be explicit, especially for Windows binary build. The current code *will* work with 1.4 or 1.6 if the arguments are changed (manually) [#15986]

4. The function XmlContainer.getNode() function has changed its signature. Instead of `XmlValue` it now returns `XmlResults`. The `XmlValue` that was previously returned can be retrieved using XmlResults.next(). There will never be more than one value in this result. It is necessary to explicitly delete the returned `XmlResults` object (XmlResults.delete()) when the application no longer needs access to the returned value. Once deleted the information in the `XmlValue` may no longer be accessible.

Python-specific Functionality Changes

1. Fixed `XmlEventReader` in Python so that methods returning unsigned char * would be mapped properly into Python strings [#15608]

2. Changed implementation of `XmlException` and related classes to make them part of the dbxml (vs _dbxml) module [#15617]

3. Changed names of `XmlException` attributes to start with lower-case letters. See src/python/README.exceptions.

4. Moved examples to dbxml/examples/python directory and added some additional basic examples

PHP-specific Functionality Changes

1. Fixed code that resulted in build and runtime errors on 64-bit platforms. One symptom was "std::bad_alloc" exceptions. The issue was a mix of 64- and 32-bit types resulting in attempts to allocate huge amounts of memory [#15587]

2. Fixed compilation problems in a threaded (ZTS) environment related to the use of incorrect macros in a few places [#15746]

3. Fixed problem (SEGV) constructing `XmlIndexLookup` objects as well as several other problems with this class implementation [#16168]

4. Moved examples to dbxml/examples/php

5. Fixed `XmlValue` constructor to accept explicitly typed strings [#15996]

Perl-specific Functionality Changes

1. Moved examples to dbxml/examples/perl

Example Code Changes

1. Added examples/cxx/xerces directory with example code that provides the same functionality that the Xerces-C DOM interfaces previously provided. They are written as example code to illustrate an integration with Xerces-C DOM and to also illustrate use of the XmlEvent* classes for such an adapter

2. Moved examples for all languages to dbxml/examples/* to consolidate them and make packaging simpler

Configuration, Documentation, Portability and Build Changes

1. XQilla 2.0 is bundled. XQilla 2.0 is released under a permissive (Apache) license

2. Windows static build projects are included

3. Project and solution files for Visual Studio version 8.00 have been added for use by Visual Studio 2005 and later releases. The new solution file is BDBXML_all_vs8.sln.

4. Added Berkeley DB project files to the BDB XML build_windows directory for Visual Studio 7.1 and 8 builds. This means that the included DB projects will be built directly in the BDB XML tree and not in the Berkeley DB tree. This does not apply to the VC6 projects and workspace and does not affect where the default build installs executables and libraries. VS7.1 project files for Berkeley DB examples are no longer included.

Berkeley DB XML 2.3.10 Change Log

Changes since Release 2.3.8 are described here. The change log of release 2.3.8 is also included, detailing the changes from 2.2.13.

Release 2.3.10 is a patch release for 2.3.8. For detailed information about the 2.3 release itself, see the 2.3.8 Change Log.

It recommended that all users move to this release. Also see the documentation included in your download package or on our website .

2.3 Release Overview

- Conformance to W3C Recommendations — XQuery 1.0 and XPath 2.0.
- Improved application integration by way of adding additional interfaces for input and output of XML content, allowing application-driven parsing and both push and pull content access without serialization of XML.
- Performance and scalability improvements.
- General bug fixes and improvements.

Changes in BDB XML 2.3.10

This section describes changes in BDB XML 2.3.10 relative to release 2.3.8.

Upgrade Requirements

Upgrade is required for containers created using release 2.2.13. See the 2.3.8 Change Log. Containers created using 2.3.8 do not require upgrade.

General Functionality Changes

1. Fixed Windows-mostly problem where dynamic_cast (RTTI) was being used by SWIG-generated code, and RTTI was not enabled in the build. The fix was to use static casting, as it's safe in these cases, rather than enabling costly RTTI. The symptom was a mysterious failure (exception thrown from the C++ runtime library) in Java when instantiating a class that extends either `XmlInputStream` or `XmlResolver` [#15280].
2. Fixed a problem where some code that should have been conditionalized based on the Berkeley DB version was not. The result is failed compilation of CompactCommand.cpp [#15283].
3. Fixed a bug where namespaces and global variables defined in the `XmlQueryContext` were not carried forward to the context of a module [#15277].
4. Fixed a bug in `XmlModify` where an assertion failure (debug mode) was triggered when removing element nodes in a certain sequence [#15291].
5. Fixed python build problems in 2.3 which caused the setup.py script to be a no-op on Unix, and even when fixed, there was a compilation problem, again Unix-specific [#15295].

6. Fix metadata indexing so that a default index on metadata will work [#15300].
7. Fixed dbxml shell to not reset the base uri in its default context. This interfered with arguments to fn:collection() [#15299].
8. Fixed a race condition in initializing function signatures. This only manifested when queries were parsed in parallel [#15298].
9. Fixed an optimisation bug that meant that using an XQuery module would sometimes cause an assert to be triggered [#15290].
10. Fixed a bug where namespaces and global variables defined in the `XmlQueryContext` were not carried forward to the context of a module [#15277].
11. Enhanced error handling in the case of opening missing or partially constructed containers [#15322].
12. Fixed a bug in `XmlModify` where some incorrect logic could lead to an incorrect query, assertion failure or SEGV. This bug would only occur after an `XmlModify` step was performed. It never occurs on the first modification step [#15321].
13. Added code to detect, and throw an exception for document/container mismatches, where a document obtained from one container is used for an update or delete operation on another container [#15320].
14. Fixed a bug that caused incorrect query plans to be generated for predicates that used the "or" operator in conjunction with indices [#15328].
15. Fixed a problem where writing an attribute with an empty value resulted in a null pointer reference (SEGV). This could happen via `XmlEventWriter`, which was used by the 2.3 upgrade code [#15335].
16. Fixed an `XmlModify` problem in 2.3.8 where inserting a new element with content would result in an exception thrown indicating that an operation was attempted on an invalid context [#15333].
17. Added -fno-strict-aliasing flag to Java library build if using gcc to avoid over-optimization of some constructs that SWIG uses for casting pointers to/from jlong. The symptom was an exception thrown indicating that there is a null or empty value [#15307].
18. Fixed a bug that resulted in the starts-with() function always using a substring index, without considering using a prefix operation on equality index [#15337].
19. Fixed an optimisation problem that caused document-uri() to fail under certain conditions [#15336].
20. Fixed the Java example modifyDocument.java to close cleanly.
21. Fixed a bug in the adding of Processing Instructions to a container.

Changes in BDB XML 2.3.8

This section describes changes in BDB XML 2.3.8 relative to release 2.2.13.

Upgrade Requirements

1. Changed database format to provide better node storage performance and scalability. Users must read documentation on container upgrade before performing an upgrade, or there is risk of data loss.

New Features

1. Conformance to Final Recommendations of XQuery 1.0 and XPath 2.0, dated January 23, 2007.

2. Use of the XQilla library replaces the use of XQuery and Pathan libraries.

3. Additional classes and related interfaces to allow direct pull and push access to XML content:

   - `XmlEventReader` — a class implementing a pull interface that enables input and output of XML content via pull events, rather than serialized XML.

   - `XmlEventWriter` — a class implementing a push interface that enables input and output of XML content via push events, rather than serialized XML.

   - `XmlEventReaderToWriter` — a class that associates an XmlEventReader with an XmlEventWriter, pushing events read from the XmlEventReader to the XmlEventWriter.

API Changes

Unless otherwise noted, the API additions apply to all language bindings, and all bindings use the same method name.

1. Added new interface classes to enable input and output of XML content via push and pull event interfaces, XmlEventReader, XmlEventWriter, and `XmlEventReaderToWriter` [#11037].

2. XmlManager:

   - getFlags() — return the flags field used to create the manager [#14403].

   - truncateContainer() and compactContainer() [#14163].

   - XmlManager::get/setImplicitTimezone() — set/get the implicit timezone for queries where it is not explicit. It is necessary to set this before doing anything with indices related to data or time types.

3. XmlContainer:

   - XmlValue getNode() — enables loading of a previously-dumped opaque (string) handle pointing directly to a node within a document in the container, dumped by XmlValue::getNodeHandle().

   - getFlags() — return the flags field used to create/open the container [#14403].

- Added `XmlContainer::putDocument()` method that takes an `XmlEventReader` object as content.
- Added `XmlContainer::putDocumentAsEventWriter()` method that allows the application to write content via an `XmlEventWriter`.

4. XmlResults:
   - getEvaluationType() — returns whether the XmlResults object is XmlQueryContext::Eager or XmlQueryContext::Lazy.
5. XmlModify:
   - Changed `XmlModify` to allow a variant that does not require an initial context of `XmlValue` or `XmlResult`. If there is only a single step, and the `XmlValue` object passed to `XmlModify::execute()` is empty (not initialized to any value), the query in that step will serve as the context for the modification [#14774].
   - Added variants of the XmlModify steps that allow specification of content as XmlResults object. This change also enables new content to comprise a sequence of elements, rather than be a single element [#14533].
6. XmlQueryContext:
   - Added `XmlQueryContext::interruptQuery()` to allow application interruption of in-process queries.
   - Added XmlQueryContext::set/getQueryTimeoutSeconds() to set timeouts on queries.
7. XmlDocument:
   - setContentAsEventReader(XmlEventReader &reader) — provide XML content as `XmlEventReader`.
   - getContentAsEventReader() — return XML content as `XmlEventReader`.
   - getContentAsEventWriter(XmlEventWriter &writer) — causes the system to write the content as events to the provided `XmlEventWriter` object.
   - Added `XmlDocument::equals` method to non-C++ APIs [#15212].
8. XmlException:
   - Added XmlException::getQueryFile, XmlException::getQueryLine() and XmlExceptions::getQueryColumn() to provide location information for XQuery errors, improving error reporting and making debugging queries easier. The reporting of this information is language binding-dependent, and may vary among the support APIs [#13465].
   - Added XmlException::ExceptionCode values:

- XmlException::EVENT_ERROR — error in XmlEvent* classes.
- XmlException::OPERATION_TIMEOUT — query operation timed out.
- XmlException::OPERATION_INTERRUPTED — query operation was interrupted.

9. XmlValue:
   - XmlValue::getTypeURI() and XmlValue::getTypeName() to return the specific type of the XmlValue returned [#14291].
   - XmlEventReader &asEventReader() — returns node content as XmlEventReader.
   - std::string getNodeHandle() — returns a string handle that represents a direct pointer to the node, and can be resolved later using XmlContainer::getNode().
10. virtual XmlInputStream *XmlResolver::resolveModuleLocation() method — allows resolution of as XQuery module namespace to a list of locations when there is no location URI.
11. virtual XmlInputStream *XmlResolver::resolveModule() method — allows resolution of XQuery modules referenced in a query.
12. Added DBXML_WELL_FORMED_ONLY flag to use to force the use of the well-formed scanner for documents. This scanner will make no attempt to retrieve schema or DTDs, even if referenced. The Java equivalent for this is XmlDocumentConfig.setWellFormedOnly() [#14055].

Changes That May Require Application Modification

1. Changed database format to provide better node storage performance and scalability [#13771].
2. Fixed a bug in the choice of index used for comparisons. Users with decimal indices may find that they need to change them to double indices before they will work correctly [#15093].
3. Changed the default base URI to be "dbxml:/" (from "dbxml:"), so that is compliant with RFC 2396 (URI specification). Containers that have been opened using an absolute unix path may now have to use aliases to reference them. See the URI rule changes below [#13881].
4. Changed how BDB XML URIs are resolved against a base URI to be conformant with the URI specification. This affects naming of containers in the arguments to fn:collection() and fn:doc(). Here are the current rules. Notes: by default, there is a base URI of "dbxml:/", and "container_alias" is either a relative path to a container, or an alias for that container, set using `XmlContainer::addAlias()`.
5. Replaced the Pathan and XQuery libraries with a single library called XQilla, which implements both XPath 2.0 and XQuery 1.0. This simplifies the build and removes redundant code [#13880].

```
New style dbxml URIs
 dbxml:/container_alias/document
 dbxml://xxx/container_alias/document (NB xxx is the URL authority,
and is ignored)
 dbxml:///container_alias/document
 dbxml:////absolute/path/to/container/document
 dbxml:/C:/windows/container/document
With base URI
 dbxml:/ + container_alias/document = dbxml:/container_alias/document
 dbxml:/container_alias + container_alias2/document2 =
dbxml:/container_alias2/document2
 dbxml:/container_alias/document + /container_alias2/document2 =
dbxml:/container_alias2/document2
 dbxml:///container_alias/document + /container_alias2/document2 =
dbxml:///container_alias2/document2
 dbxml://xxx/container_alias/document + /container_alias2/document2 =
dbxml://xxx/container_alias2/document2
 dbxml:/container_alias/document + document2 =
dbxml:/container_alias/document2
Backwards compatibility mappings
 dbxml:container_alias/document -> dbxml:/container_alias/document
```

[#14234].

6. Replaced the Pathan and XQuery libraries with a single library called XQilla, which implements both XPath 2.0 and XQuery 1.0. This simplifies the build and removes redundant code [#13880].

General Functionality Changes

1. Added several new XQuery extension functions that expose access to BDB XML functionality, as well as documentation for BDB XML XQuery extensions. New extensions are:

   - dbxml:lookup-index()
   - dbxml:lookup-attribute-index()
   - dbxml:lookup-metadata-index()
   - dbxml:handle-to-node()
   - dbxml:node-to-handle()

2. Fixed a bug where the default namespace for an element in no namespace was not redeclared, even though the element was in the middle of a document that already had a default namespace declared. [#13872]

3. Fixed a bug where a variable without a prefix was using the default namespace from the statically known namespaces [#13868]

4. Fixed bug where an attempt to open a transactional container that fails due to a version mismatch would result in errors from Berkeley DB regarding "Locker not found" and eventually an environment PANIC, requiring recovery. The same issue could arise if the open were to fail for other reasons as well [#13962].

5. Fixed a bug where using `XmlContainer::updateDocument()` on a document returned from a query that used lazy evaluation could result in an assertion failure [#13933].

6. Fixed a bug where index entries were incorrectly deleted when updating a document either via `XmlModify` or updateDocument. The problem only occurred for containers without node indices and when there were multiple, identical index entries being added/deleted during the update [#14173].

7. Fixed a problem where the Java `XmlInputStreamWrap` class, which implements java.io.InputStream did not honor the InputStream contract to return -1 at EOF. Instead it returned 0 [#14216].

8. Fixed a number of places where database errors, such as DB_ERR_DEADLOCK, were getting lost and not properly passed to the caller [#14311][#14212] [#14743][#15087].

9. Fixed a bug where errors from dbxml_dump were masked, silently resulting in partial dump files if the container had any corruption [#14388].

10. Changed dbxml_load so that it will not attempt to load into an existing container, as doing so may corrupt the container [#14381].

11. Fixed various problems in example code [#14392].

12. Fixed double-delete crash that could occur when using `XmlModify` to remove and add attributes on the same node, when the node originally has more than one attribute [#14503].

13. Fixed a problem where leading whitespace in content passed to `XmlModify` steps could result in exceptions or bad updates [#14629].

14. Fix a situation where memory could leak if an application were to use another library that includes the Xerces-C library in conjunction with BDB XML [#14451].

15. Fixed bug where node storage containers could mangle DOCYTPE declarations using PUBLIC [#14725].

16. Fixed a bug in date/time comparisons [#14949].

17. Changed index syntax names to match the documentation, as well as be case-insensitive. The old (incorrect) names (for example, year vs gYear) still work [#14961].

18. Modified index storage to be more efficient for non-string types, and fixed bugs in the ordering of index entries with certain types (for example, boolean). Also removed index types of QName and NOTATION, because namespace bindings are not available during indexing, and removed index type of anyURI, because all comparisons on anyURI values are done after casting to the string type. Upgrade will automatically replace these types with an equivalent string index [#14371].

19. Changed document ids to be 64-bit (were 32) [#15069].
20. Fixed a bug in fn:lowercase() and fn:uppercase() where they did not handle Unicode characters well [#14382].
21. Used support in Berkeley DB 4.5 to fully support CDS in BDB XML (deadlock-free, single-writer, multiple-reader concurrency, not transactional). [#14568].
22. Fixed `XmlResults::next`/previous iteration to be consistent with the semantics of the underlying object. The previous() call would not move the cursor if it was positioned on the last value in the result set [#15145].
23. Modified node allocation algorithm for node ids used in partial modifications to better utilize the node id space [#15154].
24. Fixed a problem where non-content (empty) documents could not be deleted [#15159].
25. Fixed a problem where node equality check (XmlValue::equals) would fail incorrectly when comparing across `XmlDocument` objects. Added XmlDocument::equals method to non-C++ APIs [#15212].
26. Fixed a problem where adding indices to an encrypted container would not encrypt the new index databases [#15253].
27. Added code to attempt to identify mismatched versions of Berkeley DB and BDB XML, especially in non-C++ APIs such as Java, Python, and PHP.

Utility Changes

1. Modified dbxml shell to use NULL as the default path for the DbEnv so that the DB_HOME environment variable will be used [#15139].
2. Added commands to dbxml shell:
3. setIgnore — causes it to ignore errors while not in interactive mode, allowing scripts to continue running. It has no effect on interactive sessions [#15150].
   - sync — flushes the cache to the physical databases.
   - setQueryTimeout — sets a query timeout in seconds for queries run in the shell.
   - echo — allows echoing of a string to stdout.
   - improved handling of ^C, which will now interrupt an in-progress query

Java-specific Functionality Changes

1. Fixed a Java-specific problem where using XmlDocument.setContentAsXmlInputStream() could lead to a double-delete of the underlying native memory for the `XmlInputStream` during finalization [#15126].
2. Fixed Java GettingStarted examples to use XmlContainer.delete() rather than XmlContainer.close(), and call delete() methods on all objects that need it to release container resources [#14668].

3. XmlContainer.close() is now mapped to XmlContainer.delete(), so it is no longer possible to aggressively close the underlying databases without releasing container references by explicitly deleting BDB XML objects.

Python-specific Functionality Changes

1. Added an `XmlException` class and exception hierarchy. It is described in the file, dbxml/src/python/README.exceptions [#13959].

2. Modified code generation to enable threads and correctly handle acquisition and release of the Python Global Interpreter Lock (GIL). If threads are not desired, it is possible to compile without them by modifying the source of src/python/dbxml_python_wrap.cpp to disable SWIG_PYTHON_THREADS. This can be done by defining SWIG_PYTHON_NO_THREADS [#14077].

PHP-specific Functionality Changes

1. Fixed PHP XmlManager->createContainer() and openContainer() methods to use flags arguments passed in [#14617].

2. Brought PHP interface up to date with respect to the current API, adding missing interfaces [#13889].

3. Added `XmlException` to PHP interface when compiled under PHP5, which supports exceptions. This allows PHP5 scripts to catch `XmlException` [#15245].

Tcl-specific Functionality Changes

None.

Configuration, Documentation, Portability and Build Changes

1. Bundled version 4.5.20 of Berkeley DB, which includes Multi-Version Concurrency Control (MVCC), allowing snapshot semantics for read-write concurrency. This will be quite useful for concurrent BDB XML applications (read/write concurrency).

2. XQuery and Pathan libraries have been replaced with a single library, XQilla. This library is not currently available from any other location.

3. Added dependencies to example Windows project files [#13863]. This showed up as a problem in parallel builds using Visual Studio .NET 2005.

4. Modified Unix configuration and build to allow the BDB XML library (libdbxml*) to built in a directory other than dbxml/build_unix. The buildall.sh script uses the default build locations [#14772].

5. Consolidated all projects into the single, BDBXML_all.{dsw,sln} files for Windows build, and included all Berkeley DB utilities [#14098].

Berkeley DB XML 2.3.8 Change Log

Changes since Release 2.2.13 are described here.

It recommended that all users move to this release. Also see the documentation included in your download package or on our website .

Changes in BDB XML 2.3.8

The major focus areas for release 2.3 include:

- Conformance to Proposed Recommendation of XQuery 1.0 and XPath 2.0.
- Improved application integration by way of adding additional interfaces for input and output of XML content, allowing application-driven parsing and both push and pull content access without serialization of XML.
- Performance and scalability improvements.
- General bug fixes and improvements.

Upgrade Requirements

1. Changed database format to provide better node storage performance and scalability. Users must read documentation on container upgrade before performing an upgrade, or there is risk of data loss.

New Features

1. Conformance to Proposed Recommendations of XQuery 1.0 and XPath 2.0.
2. Use of the XQilla library replaces the use of XQuery and Pathan libraries.
3. Additional classes and related interfaces to allow direct pull and push access to XML content:
4. `XmlEventReader` — a class implementing a pull interface that enables input and output of XML content via pull events, rather than serialized XML.
5. `XmlEventWriter` — a class implementing a push interface that enables input and output of XML content via push events, rather than serialized XML.
6. `XmlEventReaderToWriter` — a class that associates an `XmlEventReader` with an `XmlEventWriter`, pushing events read from the `XmlEventReader` to the `XmlEventWriter`.

API Changes

Unless otherwise noted, the API additions apply to all language bindings, and all bindings use the same method name.

1. Added new interface classes to enable input and output of XML content via push and pull event interfaces, XmlEventReader, XmlEventWriter, and `XmlEventReaderToWriter` [#11037].
2. XmlManager:

- getFlags() — return the flags field used to create the manager [#14403].
- truncateContainer() and compactContainer() [#14163].
- XmlManager::get/setImplicitTimezone() — set/get the implicit timezone for queries where it is not explicit. It is necessary to set this before doing anything with indices related to data or time types.

3. XmlContainer:
   - XmlValue getNode() — enables loading of a previously-dumped opaque (string) handle pointing directly to a node within a document in the container, dumped by `XmlValue::getNodeHandle()`.
   - getFlags() — return the flags field used to create/open the container [#14403].
   - Added an `XmlContainer::putDocument()` method that takes an `XmlEventReader` object as content.
   - Added an `XmlContainer::putDocumentAsEventWriter()` method that allows the application to write content via an `XmlEventWriter`.
4. XmlResults:
   - getEvaluationType() — returns whether the `XmlResults` object is XmlQueryContext::Eager or XmlQueryContext::Lazy.
5. XmlModify:
   - Changed `XmlModify` to allow a variant that does not require an initial context of `XmlValue` or `XmlResult`. If there is only a single step, and the `XmlValue` object passed to `XmlModify::execute()` is empty (not initialized to any value), the query in that step will serve as the context for the modification [#14774].
   - Added variants of the `XmlModify` steps that allow specification of content as `XmlResults` object. This change also enables new content to comprise a sequence of elements, rather than be a single element [#14533].
6. XmlQueryContext:
   - Added `XmlQueryContext::interruptQuery()` to allow application interruption of in-process queries.
   - Added XmlQueryContext::set/getQueryTimeoutSeconds() to set timeouts on queries.
7. XmlDocument:
   - setContentAsEventReader(XmlEventReader &reader) — provide XML content as `XmlEventReader`.
   - getContentAsEventReader() — return XML content as `XmlEventReader`.

- getContentAsEventWriter(XmlEventWriter &writer) — causes the system to write the content as events to the provided `XmlEventWriter` object.
- Added `XmlDocument::equals()` method to non-C++ APIs [#15212].

8. XmlException:
   - Added `XmlException::getQueryFile`, `XmlException::getQueryLine()` and `XmlExceptions::getQueryColumn()` to provide location information for XQuery errors, improving error reporting and making debugging queries easier. The reporting of this information is language binding-dependent, and may vary among the support APIs [#13465].
   - Added `XmlException::ExceptionCode` values:
     - XmlException::EVENT_ERROR — error in XmlEvent* classes.
     - XmlException::OPERATION_TIMEOUT — query operation timed out.
     - XmlException::OPERATION_INTERRUPTED — query operation was interrupted.
9. XmlValue:
   - `XmlValue::getTypeURI()` and `XmlValue::getTypeName()` to return the specific type of the `XmlValue` returned [#14291].
   - XmlEventReader &asEventReader() — returns node content as `XmlEventReader`.
   - std::string getNodeHandle() — returns a string handle that represents a direct pointer to the node, and can be resolved later using `XmlContainer::getNode()`
10. virtual XmlInputStream *XmlResolver::resolveModuleLocation() method — allows resolution of as XQuery module namespace to a list of locations when there is no location URI.
11. virtual XmlInputStream *XmlResolver::resolveModule() method — allows resolution of XQuery modules referenced in a query.
12. Added DBXML_WELL_FORMED_ONLY flag to use to force the use of the well-formed scanner for documents. This scanner will make no attempt to retrieve schema or DTDs, even if referenced. The Java equivalent for this is XmlDocumentConfig.setWellFormedOnly() [#14055].

Changes That May Require Application Modification

1. Changed database format to provide better node storage performance and scalability [#13771].
2. Fixed a bug in the choice of index used for comparisons. Users with decimal indices may find that they need to change them to double indices before they will work correctly [#15093].

3. Changed the default base URI to be "dbxml:/" (from "dbxml:"), so that is compliant with RFC 2396 (URI specification). Containers that have been opened using an absolute unix path may now have to use aliases to reference them. See the URI rule changes below [#13881].

4. Changed how BDB XML URIs are resolved against a base URI to be conformant with the URI specification. This affects naming of containers in the arguments to fn:collection() and fn:doc(). Here are the current rules. Notes: by default, there is a base URI of "dbxml:/", and "container_alias" is either a relative path to a container, or an alias for that container, set using `XmlContainer::addAlias()`.

```
New style dbxml URIs
 dbxml:/container_alias/document
 dbxml://xxx/container_alias/document (NB xxx is the URL
authority, and is ignored)
 dbxml:///container_alias/document
 dbxml:////absolute/path/to/container/document
 dbxml:/C:/windows/container/document

With base URI
 dbxml:/ + container_alias/document = dbxml:/container_alias/document
 dbxml:/container_alias + container_alias2/document2 =
dbxml:/container_alias2/document2
 dbxml:/container_alias/document + /container_alias2/document2 =
dbxml:/container_alias2/document2
 dbxml:///container_alias/document + /container_alias2/document2 =
dbxml:///container_alias2/document2
 dbxml://xxx/container_alias/document + /container_alias2/document2
= dbxml://xxx/container_alias2/document2
 dbxml:/container_alias/document + document2 =
dbxml:/container_alias/document2

Backwards compatibility mappings
 dbxml:container_alias/document -> dbxml:/container_alias/document
```

   [#14234].

5. Replaced the Pathan and XQuery libraries with a single library called XQilla, which implements both XPath 2.0 and XQuery 1.0. This simplifies the build and removes redundant code [#13880].

General Functionality Changes

1. Added several new XQuery extension functions that expose access to BDB XML functionality, as well as documentation for BDB XML XQuery extensions. New extensions are:

   - dbxml:lookup-index()
   - dbxml:lookup-attribute-index()

- dbxml:lookup-metadata-index()
- dbxml:handle-to-node()
- dbxml:node-to-handle()

2. Fixed a bug where the default namespace for an element in no namespace was not redeclared, even though the element was in the middle of a document that already had a default namespace declared. [#13872]

3. Fixed a bug where a variable without a prefix was using the default namespace from the statically known namespaces [#13868]

4. Fixed bug where an attempt to open a transactional container that fails due to a version mismatch would result in errors from Berkeley DB regarding "Locker not found" and eventually an environment PANIC, requiring recovery. The same issue could arise if the open were to fail for other reasons as well [#13962].

5. Fixed a bug where using `XmlContainer::updateDocument()` on a document returned from a query that used lazy evaluation could result in an assertion failure [#13933].

6. Fixed a bug where index entries were incorrectly deleted when updating a document either via `XmlModify` or updateDocument. The problem only occurred for containers without node indices and when there were multiple, identical index entries being added/deleted during the update [#14173].

7. Fixed a problem where the Java `XmlInputStreamWrap` class, which implements java.io.InputStream did not honor the InputStream contract to return -1 at EOF. Instead it returned 0 [#14216].

8. Fixed a number of places where database errors, such as DB_ERR_DEADLOCK, were getting lost and not properly passed to the caller [#14311][#14212] [#14743][#15087].

9. Fixed a bug where errors from dbxml_dump were masked, silently resulting in partial dump files if the container had any corruption [#14388].

10. Changed dbxml_load so that it will not attempt to load into an existing container, as doing so may corrupt the container [#14381].

11. Fixed various problems in example code [#14392].

12. Fixed double-delete crash that could occur when using `XmlModify` to remove and add attributes on the same node, when the node originally has more than one attribute [#14503].

13. Fixed a problem where leading whitespace in content passed to `XmlModify` steps could result in exceptions or bad updates [#14629].

14. Fix a situation where memory could leak if an application were to use another library that includes the Xerces-C library in conjunction with BDB XML [#14451].

15. Fixed bug where node storage containers could mangle DOCYTPE declarations using PUBLIC [#14725].

16. Fixed a bug in date/time comparisons [#14949].

17. Changed index syntax names to match the documentation, as well as be case-insensitive. The old (incorrect) names (for example, year vs gYear) still work [#14961].

18. Modified index storage to be more efficient for non-string types, and fixed bugs in the ordering of index entries with certain types (for example, boolean). Also removed index types of QName and NOTATION, because namespace bindings are not available during indexing, and removed index type of anyURI, because all comparisons on anyURI values are done after casting to the string type. Upgrade will automatically replace these types with an equivalent string index [#14371].

19. Changed document ids to be 64-bit (were 32) [#15069].

20. Fixed a bug in fn:lowercase() and fn:uppercase() where they did not handle Unicode characters well [#14382].

21. Used support in Berkeley DB 4.5 to fully support CDS in BDB XML (deadlock-free, single-writer, multiple-reader concurrency, not transactional). [#14568].

22. Fixed `XmlResults::next`/previous iteration to be consistent with the semantics of the underlying object. The previous() call would not move the cursor if it was positioned on the last value in the result set [#15145].

23. Modified node allocation algorithm for node ids used in partial modifications to better utilize the node id space [#15154].

24. Fixed a problem where non-content (empty) documents could not be deleted [#15159].

25. Fixed a problem where node equality check (XmlValue::equals) would fail incorrectly when comparing across `XmlDocument` objects. Added XmlDocument::equals method to non-C++ APIs [#15212].

26. Fixed a problem where adding indices to an encrypted container would not encrypt the new index databases [#15253].

27. Added code to attempt to identify mismatched versions of Berkeley DB and BDB XML, especially in non-C++ APIs such as Java, Python, and PHP.

Utility Changes

1. Modified dbxml shell to use NULL as the default path for the DbEnv so that the DB_HOME environment variable will be used [#15139].

2. Added commands to dbxml shell:

   - setIgnore — causes it to ignore errors while not in interactive mode, allowing scripts to continue running. It has no effect on interactive sessions [#15150].

   - sync — flushes the cache to the physical databases.

- setQueryTimeout — sets a query timeout in seconds for queries run in the shell.
- echo — allows echoing of a string to stdout.
- improved handling of ^C, which will now interrupt an in-progress query

Java-specific Functionality Changes

1. Fixed a Java-specific problem where using XmlDocument.setContentAsXmlInputStream() could lead to a double-delete of the underlying native memory for the `XmlInputStream` during finalization [#15126].
2. Fixed Java GettingStarted examples to use XmlContainer.delete() rather than XmlContainer.close(), and call delete() methods on all objects that need it to release container resources [#14668].
3. XmlContainer.close() is now mapped to XmlContainer.delete(), so it is no longer possible to aggressively close the underlying databases without releasing container references by explicitly deleting BDB XML objects.

Python-specific Functionality Changes

1. Added an `XmlException` class and exception hierarchy. It is described in the file, dbxml/src/python/README.exceptions [#13959].
2. Modified code generation to enable threads and correctly handle acquisition and release of the Python Global Interpreter Lock (GIL). If threads are not desired, it is possible to compile without them by modifying the source of src/python/dbxml_python_wrap.cpp to disable SWIG_PYTHON_THREADS. This can be done by defining SWIG_PYTHON_NO_THREADS [#14077].

PHP-specific Functionality Changes

1. Fixed PHP XmlManager->createContainer() and openContainer() methods to use flags arguments passed in [#14617].
2. Brought PHP interface up to date with respect to the current API, adding missing interfaces [#13889].
3. Added `XmlException` to PHP interface when compiled under PHP5, which supports exceptions. This allows PHP5 scripts to catch `XmlException` [#15245].

Tcl-specific Functionality Changes

None.

Configuration, Documentation, Portability and Build Changes

1. Bundled version 4.5.20 of Berkeley DB, which includes Multi-Version Concurrency Control (MVCC), allowing snapshot semantics for read-write concurrency. This will be quite useful for concurrent BDB XML applications (read/write concurrency).

2. XQuery and Pathan libraries have been replaced with a single library, XQilla. This library is not currently available from any other location.

3. Added dependencies to example Windows project files [#13863]. This showed up as a problem in parallel builds using Visual Studio .NET 2005.

4. Modified Unix configuration and build to allow the BDB XML library (libdbxml*) to built in a directory other than dbxml/build_unix. The buildall.sh script uses the default build locations [#14772].

5. Consolidated all projects into the single, BDBXML_all.{dsw,sln} files for Windows build, and included all Berkeley DB utilities [#14098].

Berkeley DB XML 2.2.13 Change Log

Changes since Release 2.2.8 are described here. A full list of all changes since BDB XML 2.1.8 is also available below.

It recommended that all users move to this release. Also see the documentation included in your download package or on our website .

Changes in BDB XML 2.2.13

Upgrade Requirements

None, relative to BDB XML 2.2.8, but because of the issue in [#13820] (see below) it is recommended that any containers that were:

1. used with BDB XML 2.2.8, and

2. are using node indices (DBXML_INDEX_NODES), and

3. have been modified using either `XmlModify` or `XmlContainer::updateDocument()`.

be reindexed. This can be achieved using either dbxml_dump followed by dbxml_load, or the in-place dbxml shell command, "reindexContainer."

New Features

None.

General API Changes

None.

Java-specific API changes

1. Added XmlDocument.getContentAsInputStream() to Java API to enable use of a java.io.InputStream object for document output [#13701]

2. Eliminated default constructors for Java objects [#13640]

General Functionality Changes

1. Fixed a bug where variables declared at global scope in an XQuery expression would fail with a null pointer reference [#13588]

2. Fixed bug in node storage containers where concurrent applications could receive a deadlock exception that was ignored, resulting in an assertion failure or bad pointer reference [#13597]

3. Support for CDS now requires setting of the DB_CDB_ALLDB flag if the DB_INIT_CDB flag is set in a DB environment. This is because BDB XML modifications affect more than one DB database. This may affect existing applications, which will need to change to set the DB_CDB_ALLDB flag [#13568]

4. Fixed a bug where a query using descendant-or-self() would return all named elements that follow the target node in document order, rather than just returning descendants [#13618]

5. Changed `XmlException` to DOCUMENT_NOT_FOUND when calling `XmlContainer::updateDocument()` for a document that does not exist. Previously, it would throw with the code, DATABASE_ERROR and errno, DB_NOTFOUND [#13691]

6. Fixed some problems when `DbTxn` and `XmlTransaction` were used together [#13679]

7. Fixed a bug where, if Xerces is initialized separately from BDB XML, and it outlives the last `XmlManager`, it is possible for it to contain a reference to deallocated memory [#13722]

8. Fixed a bug where adding and then deleting an index on an empty container caused an exception [#13724]

9. Fixed problem where an open of an existing container with a non-default page size using DB_CREATE would throw an exception from Berkeley DB with the message "Different pagesize specified on existent file" and errno EINVAL [#13768]

10. Fixed a bug where a crash could occur in the function, NsWriter::~NsWriter(). The problem was calling pop_back() on an empty STL vector [#13826]

11. Fixed a bug where the first result could be left out of query results. Certain ueries using the descendant axis are most vulnerable [#13816]

12. Fixed a bug where index entries could be mistakenly removed following a document update (via `XmlModify` or `XmlContainer::updateDocument`) [#13820]

13. Fixed `XmlManager::reindexContainer()`, which was introduced in BDB XML 2.2.8 to work correctly. Previously, the index type would change, but the entries created would not [#13850]

14. Fixed a bug where some node indices on attributes failed to be optimized, resulting in slower than expected queries. This only affects containers with node indices enabled [#13642]

15. Fixed a bug that occurred when a user-defined function was called from more than one place in the query, and the function used either fn:collection() or fn:doc(). This happened because a query plan was generated for the user-defined function body more than once - with the last generation replacing the former. The two query plans are now combined to make the correct query plan. [#13639]

PHP-Specific Functionality Changes

1. Added a global declaration of the `XmlIndexLookup` class that was missing from php_dbxml.cpp. This resulted in undefined references to this class that look like:

```
undefined symbol:
php_dbxml_XmlIndexLookup_ce
```

Utility Changes

1. Added a "time" sub-command to the dbxml shell utility [#13571]

Configuration, Documentation, Portability and Build Changes

1. Added support for building with Microsoft's Visual Studio 2005 [#13491]
2. Changed Windows debug information to "Program Database" consistently. This means the /Zi flag is used rather than /Z7 [#13737]
3. Changed debug .pdb filename for Pathan from PathanD.pdb to PathanD_7.1.pdb [#13737]
4. Fixed incorrect documentation regarding use of DB_DEGREE_2 in container create/open flags. The correct flag to be documented is DB_DIRTY_READ, which will be DB_READ_UNCOMMITTED in future releases that bundle DB 4.4 or higher [#13592]
5. Example code in the Getting Started Guide documentation was reviewed and updated to be correct [#13602]
6. The Getting Started Guide documentation was corrected where it erroneously stated that the XQuery expression "/Node1/Node2/Node3" would only return the first Node3 element, when it would really return all matching elements [#13604]

Changes in BDB XML 2.2.8

Upgrade Requirements

1. Changed database format from BDB XML 2.1.8 to provide better Berkeley DB Btree page fill factor. Users must read documentation on container upgrade before performing an upgrade, or there is risk of data loss [#12947]

New Features

1. Add `XmlManager::existsContainer()`, used to check for existence of a container in an efficient, non-destructive, non-intrusive manner [#11018]
2. Implement support for "fn:doc-available()" Xquery function and default collections [#12762]

3. Add `XmlIndexLookup` class to perform lookup operations, including equality, range lookups, and controlling the sort order of results (forward and reverse) [#12556]

4. Version 2.7 of the Xerces library is now used [#13177]

5. XQuery implementation has been upgraded to support the April, 2005 draft specification

General API Changes

1. Add `XmlIndexLookup` class to perform lookup operations, including equality, range lookups, and controlling the sort order of results (forward and reverse) [#12556]

2. Fix a bug where `XmlResults::hasNext()` and `XmlResults::peek()could` result in uncatchable exceptions with Lazy results [#13053]

3. Added `XmlQueryContext` get/setVariableValue variant that takes `XmlResults` to allow sequence variables [#13060]

4. Add `XmlManager::reindexContainer()` to allow an application to change the index type between node-level and document-level [#12819]

5. Add XmlManager::get/setDefaultSequenceIncrement() interfaces to allow control over the cache size of the DbSequence object used to allocate document ids [#13099]

6. Add XmlValue::BINARY as a valid XmlValue type. Add XmlValue::isBinary() interfaces to support the new type [#13221]

7. Add support for the DB_TXN_NOT_DURABLE flag when creating and opening an `XmlContainer` [#13263]

8. Add`XmlManager::openContainer()` method that takes `XmlContainerType` and mode arguments [#13285]

9. Add`XmlContainer::getIndexNodes()` to indicate if the container has node-level indices or not

General Functionality Changes

1. Modify internal data structures so that fewer lockers are required when running without transactions. This eliminates many problems that appear to be locker "leaks."[#12104]

2. Implement April 2005 drafts of XQuery 1.0 and XPath 2.0

3. Implement numerous query optimizations that should increase the query speed on all storage formats. This includes support for node-level indices specified using DBXML_INDEX_NODES [#12615]

4. Fix a bug so that unique indices will find constraint violations within a single document and between documents in a container[#12838]

5. Fix a bug where index database creation could conflict with other updates [#12839]

6. Fix a bug in QueryPlanHolder that could cause a segment fault [#13069]

7. Fix a bug during container open where the containers page size is different from the `XmlManagers` default page size [#13260]

Java-specific Functionality Changes

1. Change path to example package from com.sleepycat.dbxml.examples.gettingStarted to dbxml.gettingStarted [#12108]

2. Fix a bug so that valid DatabaseException objects are now created for all `XmlExceptions` of type DATABASE_ERROR [#12962]

3. Fix a bug so that XmlDocument.getMetaData() will correctly return binary metadata [#13193]

4. Fix a bug in `XmlInputStream` so that other classes can now be derived from this class [#13289]

Perl-specific Functionality Changes

1. Compiler information from configure is now used to set the compiler in the Perl build [#12491]

Python-specific Functionality Changes

1. Add pre-compiled Python binaries (2.4) to the Windows binary installer

2. Ship copy of the pybsddb project for convenience and ease of build

Utility Changes

1. Fix a bug in dbxml shell so that if h flag is not specified an attempt is made to join an environment in the current directory[#12993]

2. Add support for new `XmlIndexLookup` object in the dbxml shell's commands [#12556]

Configuration, Documentation, Portability and Build Changes

1. Add a version check to insure that languages such as Perl, Python, and PHP use a compatible version of Berkeley DB [#12681]

2. Change default in buildall.sh script to build thread support for Xerces platform code [#12784]

3. Add documentation for the c flag in dbxml shell [#12848]

4. Add documentation for all DatabaseConfig methods that are used by`XmlContainerConfig` that extends DatabaseConfig [#13375]

5. Add support for Berkeley DB 4.4

Berkeley DB XML 2.2.8 Change Log

Upgrade Requirements

1. Change database format to provide better Berkeley DB Btreepage fill factor. Users must read documentation on container upgrade before performing an upgrade, or there is risk of data loss. [#12947]

New Features

1. Add`XmlManager::existsContainer()`, used to check for existence of a container in an efficient, non-destructive, non-intrusive manner. [#11018]

2. Implementsupport for fn:doc-available() Xquery function and default collections.[#12762]

3. Add `XmlIndexLookup` class to perform lookup operations,including equality, range lookups, and controlling the sort order of results(forward and reverse). [#12556]

4. Version2.7 of the Xerces library is now used. [#13177]

5. XQueryimplementation has been upgraded to support the April, 2005 draft specification.

API Changes

1. Add `XmlIndexLookup` class to perform lookup operations,including equality, range lookups, and controlling the sort order of results(forward and reverse). [#12556]

2. Fix a bug where `XmlResults::hasNext()` and `XmlResults::peek()` could result in uncatchable exceptions with Lazy results. [#13053]

3. Added `XmlQueryContext` get/setVariableValue variant that takes`XmlResults` to allow sequence variables. [#13060]

4. Add `XmlManager::reindexContainer()` to allow an application tochange the index type between node-level and document-level. [#12819]

5. Add XmlManager::get/setDefaultSequenceIncrement() interfacesto allow control over the cache size of the DbSequence object used to allocatedocument ids. [#13099]

6. Add XmlValue::BINARY as a valid XmlValue type. Add XmlValue::isBinary() interfaces tosupport the new type. [#13221]

7. Add support for the DB_TXN_NOT_DURABLE flag when creating andopening an XmlContainer. [#13263]

8. Add`XmlManager::openContainer()` method that takes `XmlContainerType` and mode arguments. [#13285]

9. Add`XmlContainer::getIndexNodes()` to indicate if the container has node-level indices or not.

General Functionality Changes

1. Modify internal data structures so that fewer lockers are required when running without transactions. This eliminates many problems that appear to be locker leaks.[#12104]

2. Implement April 2005 drafts of XQuery 1.0 and XPath 2.0.

3. Implement numerous query optimizations that should increasethe query speed on all storage formats. This includes support for node-level indices specified using DBXML_INDEX_NODES [#12615]

4. Fix a bug so that unique indices will find constraint violations within a single document and between documents in a container.[#12838]

5. Fix a bug where index database creation could conflict with other updates. [#12839]

6. Fix a bug in QueryPlanHolder that could cause a segment fault. [#13069]

7. Fix a bug during container open where the containers page size is different from the XmlManager's default page size. [#13260]

Utility Changes

1. Fix a bug in dbxml shell so that if the -h flag is not specified an attempt is made to join an environment in the current directory.[#12993]

2. Add support for new `XmlIndexLookup` object in the dbxmlshell's commands. [#12556]

Java-specific Functionality Changes

1. Change path to example package from com.sleepycat.dbxml.examples.gettingStarted to dbxml.gettingStarted. [#12108]

2. Fix a bug so that valid DatabaseException objects are now created for all `XmlExceptions` of type DATABASE_ERROR. [#12962]

3. Fix a bug so that XmlDocument.getMetaData() will correctly return binary metadata. [#13193]

4. Fix a bug in `XmlInputStream` so that other classes can now be derived from this class. [#13289]

Perl-specific Functionality Changes

1. Compiler information from configure is now used to set the compiler in the Perl build. [#12491]

Python-specific Functionality Changes

1. Add pre-compiled Python binaries (2.4) to the Windows binary installer.

2. Ship copy of thepybsddb project for convenience and ease of build.

Tcl-specific Functionality Changes

None.

Configuration, Documentation, Portability and Build Changes

1. Add a version check to insure that languages such as Perl, Python, and PHP use a compatible version of Berkeley DB. [#12681]

2. Change default in buildall.sh script to build thread support for Xerces platform code. [#12784]

3. Add documentation for the -c flag in dbxml shell. [#12848]

4. Add documentation for all `DatabaseConfig` methods that are used by `XmlContainerConfig` that extends DatabaseConfig. [#13375]

5. Add support for Berkeley DB 4.4.

Berkeley DB XML 2.1.7 & 2.1.8 Change Log

Upgrade Requirements

1. The database format has changed in order to fix a platform portability problem. To upgrade your DB XML containers to 2.1.8, please follow the steps described in Upgrading Berkeley DB XML Applications to 2.1 (page 37).

2. DB XML 2.0 applications will need to be recompiled and relinked using the DB XML 2.1 libraries.

3. DB XML 2.1.8 is a patch release for DB XML 2.1.7.

New Features

1. Add the Perl API.

2. Add Windows binaries and installer.

3. A number of enhancements to the dbxml command line shell utility, including renaming it from "dbxml_shell" to "dbxml"

4. Add an "Introducing Berkeley DB XML" document to help newcomers quickly get started using the product.

API Changes

1. Add XmlContainer::{add,remove}Alias() to allow applications to create aliases for containers for use in XQuery expressions. [#11715]

2. Add `XmlContainer::getAllDocuments()` method to return all documents in a container.

3. Add `XmlContainer::getNumDocuments()` method to return the number of documents in a container.

4. Implement a functioning `XmlManager::upgradeContainer()` interface to upgrade containers from 2.0.9 to 2.1.8.

5. Change the default Base URI to "dbxml:" (it was "dbxml:/"). [#11715]

6. Change resolution of "dbxml:" URIs to treat a leading slash ("dbxml:/path" or "dbxml:C:/") as an absolute path. Previously, all paths were treated as relative inside an XQuery expression. [#11715]

General Functionality Changes

1. Fix a performance problem with Pathan StringPool objects when returning large data sets. [#12532]

2. Fix a bug where an entity in an attribute value was not being expanded during document serialization if a child node had been added to that element via direct DOM manipulation or XmlModify. [#12489]

3. Fix a bug where a DOM assertion failure would occur when asking for a specific item from a DOM nodelist. [#12481]

4. Change all C++ objects so that they now have default constructors to make it easier to use them as class members. If these uninitialized default objects are used an exception will be thrown.

5. Fix a bug where removal of metadata in a document retrieved using the DBXML_LAZY_DOCS flag would fail. [#12369]

6. Fix a bug where a virtual collection created by `XmlResolver::resolveCollection` would block or potentially crash after return. [#12154]

7. Fix a bug that caused containers to be non-portable across machine architectures (big vs little endian). Part of this fix causes index databases to be created on demand, resulting in significantly smaller initial container size. [#12196]

8. Fix a bug where default settings could result in creating a container that stored whole documents, rather than the intended default of node storage. [#12193]

9. Fix a bug where entities were not being escaped in attribute values returned in serialized query results from whole document storage. [#12193]

10. Fix bugs in the DOMDocument::getEncoding() and DOMDocument::getActualEncoding() methods. [#11956]

11. Fix bug where it was possible to remove the implicit document name index and remove the name metadata item from `XmlDocument` objects. [#11948]

12. Fix a bug where substring indices on 3-character values would not not properly lower-case the values for storage. This resulted in failed queries where they should have succeeded. [#11945]

13. Fix a memory leak in NsEventGenerator where removal of certain node storage documents leaked an 8k buffer. [#11857]

14. Fix a bug where `XmlContainer::updateDocument` could not be called on a node storage document obtained from a query using `XmlQueryContext::EvaluationType` of Lazy. This is now legal. [#11838]

15. Fix a bug in the query planner that could result in NULL query plan objects and cause the BDB XML application to crash. [#11831]

16. Fix a bug where serialization of constructed DOM in queries was not properly tracking and expanding entities. [#11817]

17. Change whole document storage to only validate documents on initial input, and not for querying or removal. This only affects containers for whole document storage that have validation enabled. [#11809]

18. Change Pathan to not use Windows HeapAlloc, etc. for its MemoryManager implementation. It can cause memory shortages in certain queries. [#11806]

19. Fix a bug where empty elements in node storage documents could result in using the wrong namespace prefixes for subsequent elements. [#11762]

20. Fix a bug where `XmlResults::previous` would crash on an empty result set. Fixed several related `XmlResults` iterator issues. [#11756]

21. Fix a bug where BDB XML didn't allow a query to operate directly against `XmlInputStream` in an XmlDocument. [#11755]

22. Fix a bug where BDB XML could crash on queries that asked for text of attribute notes (for example, /foo/@attr/text()). [#11717]

23. Fix a bug where the isSpecified() state was not being tracked properly for attributes. This would appear when using the DOM or when serializing elements with unspecified attributes. [#11686]

24. Fix a bug in the database version checking so that it runs before any other operations and correctly reports database version mismatch errors on BDB XML 1.x version containers. [#11619]

Utility Changes

1. Change the name of the dbxml_shell utility to be simply dbxml. [#12226]

2. Change dbxml shell command names to be more obvious. Enhance dbxml shell help and usage messages. Compatibility with 2.0.9 command names is maintained where possible. [#11610]

3. Change the argument flags to the utility commands so that they are consistent. Some utility command flags will have been modified from 2.0.9.

Java-specific Functionality Changes

1. Fix `XmlContainer.putDocument()` methods to return `String`names of new documents. [#12006]

2. Java interfaces are defined to only throw XmlException. Implement `XmlException.getDatabaseException()` as documented. [#12377]

Python-specific Functionality Changes

1. Fix a bug where the default container type was not NodeContainer. [#11625]

2. Fix a bug where underlying `XmlContainer` objects created by `XmlManager.createContainer()` were not getting deleted correctly. [#11627]

Perl-specific Functionality Changes

1. Fix a problem where a crash could occur if an object created by an `XmlManager` outlived the `XmlManager` itself (no SR #).

Tcl-specific Functionality Changes

1. Fix a bug where the default container type was not NodeContainer. [#11625]

2. Fix a bug where underlying `XmlContainer` objects created by XmlManager.createContainer() were not getting deleted correctly. [#11627]

Configuration, Documentation, Portability and Build Changes

1. Fix a bug that caused gcc 4.0 builds to fail, where NsXercesDom functions used an incorrect virtual function table because of object confusion. [#12527]

2. Fix a bug in the simpleAdd.cpp example where it was not setting DBXML_GEN_NAME on putDocument calls. [#11796]

3. Add configuration and build support for native compilers on HP-UX (aCC) and AIX (xlC_r). [#11777]

4. Add build support for the Cygwin platform on Windows.

5. Add configuration support for the Fedora Core 3 platform. [#11650]

6. Add configuration support so that on Unix platforms the configure script will use its parameters to create setup.py for the Python build. [#11541]

7. Add the Perl API.

8. The destination for Windows build artifacts, including libraries, executables, JAR files, and header files has been changed. They are now placed in the directories, "bin," "lib," "include," and "jar" off of the top-level installation directory.

9. Add Windows binaries and installer.

10. Add a number of enhancements to the dbxml command line shell utility.

11. Add an "Introducing Berkeley DB XML" document to help newcomers quickly get started using the product.

Berkeley DB XML 2.0.7 & 2.0.9 Change Log

Upgrade Requirements

1. The database format has changed due to the large number of new features and new storage and indexing options in in 2.0. These changes are incompatible with DB XML 1.2.1 databases. Consequently applications will need to manually dump any existing DB XML 1.2.1 databases and load them into a new DB XML 2.0 database or reload them from the source XML documents.

New Features

1. Support for XPath 2.0, including new data types.

2. Support for XQuery 1.0.

3. Support for additional index types.

4. Improved large document support.

5. Significant API improvement and extension to support new functions. Introduction of factory methods to create objects.

6. Support for fine grained document storage.

7. Naming and cross-container operation support.

8. Enhanced Metadata support.

9. Support for document loading. Documents can now be loaded from input streams (including local files and URLs) or raw memory locations.

10. Enhanced document update functionality.

11. Support for DOM-like navigation methods on XmlValue.

12. New classes for `XmlInputStream`, `XmlData`, `XmlManager`, `XmlStatistics`, `XmlMetaDataIterator` and `XmlTransaction`.

General Functionality Changes

1. Fix a bug where `XmlContainer::getDocument()` was not passing in user supplied flags. [#9607]

2. Fix a bug where a Container::Close() did not reset ID members, so a re-create failed to write them to a new container. Fix a bug where open() and versionCheck() would attempt to write even when the flags specified DB_READONLY. [#9608]

3. Fix a bug where duplicate index keys were not sorted, causing poor performance for deleteDocument(). [#9737]

4. Fix a bug where adding an index to a container which contains documents in a CDS configured environment would cause the thread to self-deadlock. [#9925]

5. Fix a bug where documents weren't being returned for queries on the presence of metadata when the metadata was indexed. [#10487]

6. Fix a double-free bug in Node Storage for certain size/shaped documents. [#11641]

API Changes

1. New classes for `XmlInputStream`, `XmlData`, `XmlManager`, `XmlStatistics`, `XmlMetaDataIterator` and `XmlTransaction`.

2. Extensive changes to `XmlModify`.

Java-specific API Changes

1. Add `Environment` and `Transaction` objects.

2. Add `XmlManagerConfig`, `XmlContainerConfig` and `XmlDocumentConfig` objects.

Python-specific API Changes

None.

PHP-specific API Changes

1. Add a third party supported PHP API to the distribution, including enhanced support for Apache integration.

Tcl-specific API Changes

None.

Utility Changes

1. Add dbxml_shell utility to allow direct interaction with DB XML containers, indices and documents.

2. Add dbxml_load_container utility to facilitate initial loading of documents into a container.

Configuration, Documentation, Portability and Build Changes

1. Simplify installation to include all required packages.

2. Support for Berkeley DB 4.3.

3. Fixed bugs in header files on Linux Fedora Core 3 installation. [#11650]

Berkeley DB XML 1.2.1 Change Log

Note

Release 1.2.1 is a bug fix release for 1.2.0.

Upgrade Requirements

None.

New Features

1. Added PHP API with limited support, documentation and testing. For more information please see the src/php4/README in your download package.

General Functionality Changes

1. Fix a bug in the query processor where queries that combined an equality and a presence lookup, in that order, would not optimize away the presence lookup. [#9437]

2. Fix a bug where substring queries for values with more than five characters caused an assertion failure. [9504]

3. Fix a bug where a substring search for 3-character string did not normalize the string. [#9573]

API Changes

None.

Java-specific API Changes

None.

Python-specific API Changes

1. Fix a bug where Python failed to find a function overload. This occurred in all overloaded `XmlContainer` methods that take a DbTxn * as their first argument, notable `XmlContainer::deleteDocument()` and `XmlContainer::queryWithXPath()`. This problem would only occur in release 1.2.0 [#9443]

Tcl-specific API Changes

None.

Utility Changes

None.

Configuration, Documentation, Portability and Build Changes

1. Fix a bug where the use of std::compare was not supported by gcc 2.9x. [#9526]

Berkeley DB XML 1.2.0 Change Log

Upgrade Requirements

1. The container format has changed. Containers created with version 1.1.0 of Berkeley DB XML must be upgraded by calling the `XmlContainer::upgrade()` method. Attempting to open a 1.1.0 version container with the 1.2.0 library will return a VERSION_MISMATCH exception.

New Features

1. New API to perform in-place modification of documents, based on query results has been added. It addes `XmlContainer::modifyDocument()` and `XmlDocument::modifyDocument()`, along with a new class to specify the modification, XmlModify.

General Functionality Changes

1. Fix a query planner bug that would generate an incorrect plan when edge indices were enabled and the query included the text() function. Queries such as '/a/b/text()', where b is edge indexed, would generate a plan looking for edges a.b and b.b. [#8527]

2. Fix a query planner bug where it tried to map an XPath equality sub-expression onto an appropriate equality index, but if not avaliable did not look for a suitable presence index instead. [#8566]

3. Fix a bug where variable references in parsed query expressions only worked the first time. Each subsequent use of the expression would fail to make use of the current value defined in the query context. [#8585]

4. Fix a bug where the XPath expression parser did not allow node names to have axis names, for example child, parent, etc. [#8781]

5. Fix a bug where queries of the form /a[b/c='foo'], where 'c' is indexed, always resulted in a sequential scan instead of an index lookup. [#8819]

6. Fix a bug where XPath queries of the form /ns:div and /ns:mod were not supported. [#8840]

7. Modify XPath functions substring-before and substring-after to make use of a substring index. [#8875]

8. Modify XPath function starts-with to make use of a substring index, if an equality index is not available. [#8875]

9. Remove support for `XmlValue` types NODELIST and ATTRIBUTE. They have been replaced with the type NODE. An XPath query that matches a list of nodes within a document will

now return a list of `XmlValue` objects with type NODE. The `XmlValue` methods isNodeList, isAttribute, asNodeList, and asAttribute, have been replaced by isNode and asNode. [#8884]

10. Substring indices are now created using a normalized node value. All whitespace and punctuation characters are removed, and all upper case characters are converted to their lower case equivalents, except UTF-8 characters which are not changed. Each key is now three characters rather than three bytes, as a UTF-8 character can theoreticaly occupy as many as six bytes. Queries against substring indices will still return the same results. This change improves the performance of the indexer and reduces the storage space requiements of the substring index. The container format has changed meaning that containers created with version 1.1.0 of Berkeley DB XML must be upgraded by calling the `XmlContainer::upgrade()` method. [#8906]

11. Fix a bug where the namespace prefix was not stored with document metadata items. This had the consequence of the metadata item not being reflected into the document when retrieved as a DOM. [#8967]

12. Fix a bug where numeric comparisons were precision limited. [#8967]

13. Fix a bug where QueryContext should have been holding a reference to Xerces. [#9050]

14. Fix the query processor to correctly identify opportunities to make use of edge indices. Queries such as /a[b and c] would not make use of edge indices for a.b or a.c. [#9009] [#9334]

API Changes

1. Add `XmlDocument::queryWithXPath(XmlQueryExpression...)`. Previously available on XmlContainer, but not XmlDocument. [#8591]

2. Add an option to `XmlDocument::getDOM()` for adoption of the contained DOM. [#8229]

3. Remove support for `XmlValue` types NODELIST and ATTRIBUTE. They have been replaced with the type NODE. An XPath query that matches a list of nodes within a document will now return a list of `XmlValue` objects with type NODE. The `XmlValue` methods isNodeList, isAttribute, asNodeList, and asAttribute, have been replaced by isNode and asNode. [#8884]

4. DB_CHKSUM_SHA1 is now DB_CHECKSUM.

5. The DbTxn argument has been removed from the `XmlResults::next()` iterators. It is not needed. Compatibility has been maintained for C++, but not for Java and scripting languages; some code may need to be changed. [#8903]

6. New API to perform in-place modification of documents, based on query results has been added. It addes `XmlContainer::modifyDocument()` and `XmlDocument::modifyDocument()`, along with a new class to specify the modification, XmlModify. The operations available are intended to provide the same update capabilities as those specified by XUpdate , except via API rather than an XML language. This API

is not an implementation of the XML::DB API, but could be used as a basis for such an implementation.

7. `XmlQueryContext::setWithMetaData(bool)` has been added to allow an application to query a document, or documents, without including the document metadata as attributes on the root node. There is cost associated with adding metadata as attributes, and if they are not required for correctness of a query, it is more efficient to leave them out. The default, for compatibility, is to include the metadata in queries. There is a corresponding getWithMetaData() function, as well.

Java-specific API Changes

1. Fix memory leaks. [#8555]
2. `XmlContainer::queryWithXPathExpression` renamed to `XmlContainer::queryWithXPath`.
3. `XmlContainer::deleteByID` renamed to `XmlContainer::deleteDocument`.
4. `XmlResults::close` removed in favor of `XmlResults::delete`.

Python-specific API Changes

1. Fix memory leaks. [#8555]

Tcl-specific API Changes

1. Fix memory leaks. [#8555]

Utility Changes

None.

Configuration, Documentation, Portability and Build Changes

1. The XPath union operator bug has been fixed in Pathan 1.2 release 2. [#8528]
2. Fix test suite to pass in transactional mode and to work against Xerces-C++ version 2.3. [#8679]
3. Improve configure to deal with solaris configurations better. [#8810]
4. Fix creation of JNI library for Mac OS/X with the required extension of .jnilib. [#9013]
5. Test with Berkeley DB 4.2.52.
6. Test with Xerces 2.3 from download, Xerces 2.3 from CVS, and Xerces 2.4.
7. Test with Pathan 1.1 release 2.
8. Update DB XML Getting Started Guide for C++.

9. Add DB XML Getting Started Guide for Java.

Known Issues

1. Perl build on OS X may fail. If you run into this (or fix it!), please let Sleepycat know.

2. Gcc 3.3 — if you are using gcc 3.3, the build of Pathan1.2, release 2 may fail. If so, it's necessary to get the code directly from the CVS tree.

3. There is a bug in Python's use of overloaded functions in the class, XmlContainer, specifically, those that take a DbTxn * as the first argument — deleteDocument() and queryWithXPath(). The error is a failure to find a matching function overload. It only occurs when the DbTxn is other than "None." This is fixed, and will be released in 1.2.1.