

Oracle Berkeley DB

Upgrade Guide

12c Release 1
Library Version 12.1.6.1



Legal Notice

This documentation is distributed under an open source license. You may review the terms of this license at: <http://www.oracle.com/technetwork/database/berkeleydb/downloads/oslicense-093458.html>

Oracle, Berkeley DB, and Sleepycat are trademarks or registered trademarks of Oracle. All rights to these marks are reserved. No third-party use is permitted without the express prior written consent of Oracle.

Other names may be trademarks of their respective owners.

To obtain a copy of this document's original source code, please submit a request to the Oracle Technology Network forum at: <https://forums.oracle.com/forums/forum.jspa?forumID=271>

Published 7/26/2017

Table of Contents

Preface	xi
Conventions Used in this Book	xi
For More Information	xi
Contact Us	xii
1. Introduction	1
Library version information	1
2. Upgrading from previous versions of Berkeley DB	2
3. Upgrading Berkeley DB 4.6 applications to Berkeley DB 4.7	6
Introduction	6
Run-time configuration	6
Replication API	6
Tcl API	6
DB_ENV->set_intermediate_dir	7
Log configuration	7
Upgrade Requirements	7
Berkeley DB 4.7.25 Change Log	7
Database or Log File On-Disk Format Changes:	7
New Features:	7
Database Environment Changes:	8
Concurrent Data Store Changes:	8
General Access Method Changes:	8
Btree Access Method Changes:	9
Hash Access Method Changes:	9
Queue Access Method Changes:	9
Recno Access Method Changes:	9
C-specific API Changes:	9
Java-specific API Changes:	9
Direct Persistence Layer (DPL), Bindings and Collections API:	10
Tcl-specific API Changes:	10
RPC-specific Client/Server Changes:	11
Replication Changes:	11
XA Resource Manager Changes:	12
Locking Subsystem Changes:	12
Logging Subsystem Changes:	12
Memory Pool Subsystem Changes:	13
Mutex Subsystem Changes:	13
Transaction Subsystem Changes:	13
Utility Changes:	13
Configuration, Documentation, Sample Application, Portability and Build Changes:	14
4. Upgrading Berkeley DB 4.5 applications to Berkeley DB 4.6	15
Introduction	15
C API cursor handle method names	15
DB_MPOOLFILE->put	15
B_MPOOLFILE->set	15
Replication Events	16

DB_REP_FULL_ELECTION	16
Verbose Output	16
DB_VERB_REPLICATION	17
Windows 9X	17
Upgrade Requirements	17
Berkeley DB 4.6.21 Change Log	17
4.6.21 Patches:	17
4.6.19 Patches	18
Database or Log File On-Disk Format Changes:	18
New Features:	18
Database Environment Changes:	19
Concurrent Data Store Changes:	20
General Access Method Changes:	20
Btree Access Method Changes:	21
Hash Access Method Changes:	21
Queue Access Method Changes:	21
Recno Access Method Changes:	21
C++-specific API Changes:	21
Java-specific API Changes:	21
Java collections and bind API Changes:	22
Tcl-specific API Changes:	22
RPC-specific Client/Server Changes:	22
Replication Changes:	22
XA Resource Manager Changes:	24
Locking Subsystem Changes:	24
Logging Subsystem Changes:	24
Memory Pool Subsystem Changes:	24
Transaction Subsystem Changes:	24
Utility Changes:	24
Configuration, Documentation, Portability and Build Changes:	25
5. Upgrading Berkeley DB 4.4 applications to Berkeley DB 4.5	26
Introduction	26
deprecated interfaces	26
DB->set_isalive	26
DB_ENV->rep_elect	26
Replication method naming	27
Replication events	27
Memory Pool API	27
DB_ENV->set_paniccall	27
DB->set_pagesize	27
Collections API	28
--enable-pthread_self	28
Recno backing text source files	28
Application-specific logging	28
Upgrade Requirements	29
Berkeley DB 4.5.20 Change Log	29
Database or Log File On-Disk Format Changes:	29
New Features:	29
Database Environment Changes:	29

Concurrent Data Store Changes:	30
General Access Method Changes:	30
Btree Access Method Changes:	30
Hash Access Method Changes:	31
Queue Access Method Changes:	31
Recno Access Method Changes:	31
C++-specific API Changes:	31
Java-specific API Changes:	31
Java collections and bind API Changes:	31
Tcl-specific API Changes:	31
RPC-specific Client/Server Changes:	32
Replication Changes:	32
XA Resource Manager Changes:	32
Locking Subsystem Changes:	32
Logging Subsystem Changes:	32
Memory Pool Subsystem Changes:	33
Transaction Subsystem Changes:	33
Utility Changes:	34
Configuration, Documentation, Portability and Build Changes:	34
6. Upgrading Berkeley DB 4.3 applications to Berkeley DB 4.4	35
Introduction	35
DB_AUTO_COMMIT	35
DB_DEGREE_2, DB_DIRTY_READ	35
DB_JOINENV	35
mutexes	36
DB_MPOOLFILE->set_clear_len	36
lock statistics	37
Upgrade Requirements	37
Berkeley DB 4.4.16 Change Log	37
Database or Log File On-Disk Format Changes:	37
New Features:	37
Database Environment Changes:	38
Concurrent Data Store Changes:	38
General Access Method Changes:	39
Btree Access Method Changes:	39
Hash Access Method Changes:	40
Queue Access Method Changes:	40
Recno Access Method Changes	40
C++-specific API Changes:	40
Java-specific API Changes:	40
Java collections and bind API Changes:	41
Tcl-specific API Changes:	42
RPC-specific Client/Server Changes:	42
Replication Changes:	42
XA Resource Manager Changes:	43
Locking Subsystem Changes:	43
Logging Subsystem Changes:	43
Memory Pool Subsystem Changes:	44
Transaction Subsystem Changes:	44

Utility Changes:	45
Configuration, Documentation, Portability and Build Changes:	45
Berkeley DB 4.4.20 Change Log	46
Changes since Berkeley DB 4.4.16:	46
7. Upgrading Berkeley DB 4.2 applications to Berkeley DB 4.3	47
Introduction	47
Java	47
DB_ENV->set_errcall, DB->set_errcall	48
DBCursor->c_put	48
DB->stat	48
DB_ENV->set_verbose	48
Logging	49
DB_FILEOPEN	49
ENOMEM and DbMemoryException	49
Replication	49
Run-time configuration	50
Upgrade Requirements	50
Berkeley DB 4.3.29 Change Log	50
Database or Log File On-Disk Format Changes:	50
New Features:	50
Database Environment Changes:	50
Concurrent Data Store Changes:	52
General Access Method Changes:	52
Btree Access Method Changes:	53
Hash Access Method Changes:	53
Queue Access Method Changes:	54
Recno Access Method Changes	54
C++-specific API Changes:	54
Java-specific API Changes:	55
Tcl-specific API Changes:	55
RPC-specific Client/Server Changes:	56
Replication Changes:	56
XA Resource Manager Changes:	57
Locking Subsystem Changes:	57
Logging Subsystem Changes:	58
Memory Pool Subsystem Changes:	58
Transaction Subsystem Changes:	58
Utility Changes:	59
Configuration, Documentation, Portability and Build Changes:	60
8. Upgrading Berkeley DB 4.1 applications to Berkeley DB 4.2	62
Introduction	62
Java	62
Queue access method	63
DB_CHKSUM_SHA1	64
DB_CLIENT	64
DB->del	64
DB->set_cache_priority	64
DB->verify	64
DB_LOCK_NOTGRANTED	65

Replication	65
Replication initialization	65
Database methods and replication clients	65
DB_ENV->rep_process_message()	66
Client replication environments	66
Tcl API	66
Upgrade Requirements	66
Berkeley DB 4.2.52 Change Log	66
Database or Log File On-Disk Format Changes:	66
New Features:	66
Database Environment Changes:	67
Concurrent Data Store Changes:	69
General Access Method Changes:	69
Btree Access Method Changes:	70
Hash Access Method Changes:	71
Queue Access Method Changes:	71
Recno Access Method Changes:	72
C++-specific API Changes:	73
Java-specific API Changes:	73
Tcl-specific API Changes:	74
RPC-specific Client/Server Changes:	75
Replication Changes:	75
XA Resource Manager Changes:	78
Locking Subsystem Changes:	78
Logging Subsystem Changes:	79
Memory Pool Subsystem Changes:	80
Transaction Subsystem Changes:	80
Utility Changes:	81
Configuration, Documentation, Portability and Build Changes:	81
9. Upgrading Berkeley DB 4.0 applications to Berkeley DB 4.1	84
Introduction	84
DB_EXCL	84
DB->associate, DB->open, DB->remove, DB->rename	84
DB_ENV->log_register	86
st_flushcommit	86
DB_CHECKPOINT, DB_CURLSN	86
DB_INCOMPLETE	87
DB_ENV->memp_sync	87
DB->stat.hash_nelem	87
Java exceptions	87
C++ exceptions	87
Application-specific logging and recovery	88
Upgrade Requirements	88
Berkeley DB 4.1.24 and 4.1.25 Change Log	88
Database or Log File On-Disk Format Changes:	88
Major New Features:	88
General Environment Changes:	89
General Access Method Changes:	90
Btree Access Method Changes:	91

Hash Access Method Changes:	91
Queue Access Method Changes:	92
Recno Access Method Changes:	92
C++-specific API Changes:	92
Java-specific API Changes:	93
Tcl-specific API Changes:	93
RPC-specific Client/Server Changes:	93
Replication Changes:	93
XA Resource Manager Changes:	93
Locking Subsystem Changes:	94
Logging Subsystem Changes:	94
Memory Pool Subsystem Changes:	94
Transaction Subsystem Changes:	95
Utility Changes:	95
Configuration, Documentation, Portability and Build Changes:	95
Berkeley DB 4.1.25 Change Log	97
10. Upgrading Berkeley DB 3.3 applications to Berkeley DB 4.0	98
Introduction	98
db_deadlock	98
lock_XXX	98
log_XXX	98
memp_XXX	99
txn_XXX	100
db_env_set_XXX	101
DB_ENV->set_server	102
DB_ENV->set_lk_max	102
DB_ENV->lock_id_free	102
Java CLASSPATH environment variable	102
C++ ostream objects	103
application-specific recovery	103
Upgrade Requirements	104
4.0.14 Change Log	104
Major New Features:	104
General Environment Changes:	104
General Access Method Changes:	105
Btree Access Method Changes:	105
Hash Access Method Changes:	105
Queue Access Method Changes:	105
Recno Access Method Changes:	105
C++ API Changes:	105
Java API Changes:	105
Tcl API Changes:	105
RPC Client/Server Changes:	106
XA Resource Manager Changes:	106
Locking Subsystem Changes:	106
Logging Subsystem Changes:	106
Memory Pool Subsystem Changes:	106
Transaction Subsystem Changes:	107
Utility Changes:	107

Database or Log File On-Disk Format Changes:	107
Configuration, Documentation, Portability and Build Changes:	107
11. Upgrading Berkeley DB 3.2 applications to Berkeley DB 3.3	109
introduction	109
DB_ENV->set_server	109
DB->get_type	109
DB->get_byteswapped	109
DB->set_malloc, DB->set_realloc	109
DB_LOCK_CONFLICT	110
memp_fget, EIO	110
txn_prepare	111
--enable-dynamic, --enable-shared	111
--disable-bigfile	111
Upgrade Requirements	111
12. Upgrading Berkeley DB 3.1 applications to Berkeley DB 3.2	112
introduction	112
DB_ENV->set_flags	112
DB callback functions, app_private field	112
Logically renumbering records	112
DB_INCOMPLETE	113
DB_ENV->set_tx_recover	113
DB_ENV->set_mutexlocks	113
Java and C++ object reuse	114
Java java.io.FileNotFoundException	114
db_dump	114
Upgrade Requirements	114
13. Upgrading Berkeley DB 3.0 applications to Berkeley DB 3.1	115
introduction	115
DB_ENV->open, DB_ENV->remove	115
DB_ENV->set_tx_recover	115
DB_ENV->set_feedback, DB->set_feedback	115
DB_ENV->set_paniccall, DB->set_paniccall	116
DB->put	116
identical duplicate data items	117
DB->stat	117
DB_SYSTEM_MEM	117
log_register	118
memp_register	118
txn_checkpoint	118
environment configuration	118
Tcl API	119
DB_TMP_DIR	119
log file pre-allocation	119
Upgrade Requirements	120
14. Upgrading Berkeley DB 2.X applications to Berkeley DB 3.0	121
introduction	121
environment open/close/unlink	121
function arguments	124
DB_ENV structure	125

database open/close	126
db_xa_open	127
DB structure	127
DBINFO structure	128
DB->join	129
DB->stat	130
DB->sync and DB->close	130
lock_put	130
lock_detect	130
lock_stat	130
log_register	130
log_stat	130
memp_stat	131
txn_begin	131
txn_commit	131
txn_stat	131
DB_RMW	131
DB_LOCK_NOTHELD	131
EAGAIN	132
EACCES	132
db_jump_set	132
db_value_set	133
DbEnv class for C++ and Java	133
Db class for C++ and Java	135
additional C++ changes	135
additional Java changes	135
Upgrade Requirements	136
15. Upgrading Berkeley DB 1.85 or 1.86 applications to Berkeley DB 2.0	137
Introduction	137
System Integration	137
Converting Applications	138
Upgrade Requirements	139

Preface

Welcome to Berkeley DB (DB). This document describes how to upgrade from previous versions of Berkeley DB.

This document reflects Berkeley DB 12c Release 1, which provides DB library version 12.1.6.1.

Conventions Used in this Book

The following typographical conventions are used within in this manual:

Structure names are represented in monospaced font, as are method names. For example: "DB->open() is a method on a DB handle."

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

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

```
/* File: gettingstarted_common.h */
typedef struct stock_dbs {
    DB *inventory_dbp; /* Database containing inventory information */
    DB *vendor_dbp;    /* Database containing vendor information */

    char *db_home_dir; /* Directory containing the database files */
    char *inventory_db_name; /* Name of the inventory database */
    char *vendor_db_name; /* Name of the vendor database */
} STOCK_DBS;
```

Note

Finally, notes of interest are represented using a note block such as this.

For More Information

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

- [Berkeley DB Installation and Build Guide](#)
- [Getting Started with Transaction Processing for C](#)
- [Berkeley DB Getting Started with Replicated Applications for C](#)
- [Berkeley DB C API Reference Guide](#)
- [Berkeley DB C++ API Reference Guide](#)
- [Berkeley DB STL API Reference Guide](#)

-
- [Berkeley DB TCL API Reference Guide](#)
 - [Berkeley DB Programmer's Reference Guide](#)
 - [Berkeley DB Getting Started with the SQL APIs](#)

To download the latest Berkeley DB 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 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 at: <https://forums.oracle.com/forums/forum.jspa?forumID=271>, or for Oracle Berkeley DB High Availability at: <https://forums.oracle.com/forums/forum.jspa?forumID=272>.

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

This manual describes how to upgrade from historical versions of Berkeley DB (Berkeley DB 4.7 and older). For information on upgrading newer releases of the product (anything newer than DB 4.7), and on building and installing Berkeley DB on all of the platforms it officially supports, see the [Berkeley DB Installation and Build Guide](#).

Library version information

Each release of the Berkeley DB 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 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 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 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 version information, applications should use the `DB_ENV->version()` function. In addition to the previous information, the `DB_ENV->version()` function returns a string encapsulating the version information, suitable for display to a user.

Chapter 2. Upgrading from previous versions of Berkeley DB

The following information describes the general process of upgrading Berkeley DB installations. There are four areas to be considered when upgrading Berkeley DB applications and database environments: the application API, the database environment's region files, the underlying database formats, and, in the case of transactional database environments, the log files. The upgrade procedures required depend on whether or not the release is a major or minor release (in which either the major or minor number of the version changed), or a patch release (in which only the patch number in the version changed). Berkeley DB major and minor releases may optionally include changes in all four areas, that is, the application API, region files, database formats, and log files may not be backward-compatible with previous releases.

Each Berkeley DB major or minor release described in this book has a chapter indicating how to upgrade to the new release. The chapter describes any API changes made in the release. Application maintainers should review the API changes and update their applications as necessary before recompiling with the new release. In addition, each chapter includes a section specifying whether the log file format or database formats changed in non-backward-compatible ways as part of the release. Because there are several underlying Berkeley DB database formats, and they do not all necessarily change in the same release, changes to a database format in a release may not affect any particular application. Further, database and log file formats may have changed but be entirely backward-compatible, in which case no upgrade will be necessary.

A Berkeley DB patch release will never modify the API, regions, log files, or database formats in incompatible ways, and so applications need only be relinked (or, in the case of a shared library, pointed at the new version of the shared library) to upgrade to a new release. Note that internal Berkeley DB interfaces may change at any time and in any release (including patch releases) without warning. This means the library must be entirely recompiled and reinstalled when upgrading to new releases of the library because there is no guarantee that modules from one version of the library will interact correctly with modules from another release. We recommend using the same compiler release when building patch releases as was used to build the original release; in the default configuration, the Berkeley DB library shares data structures from underlying shared memory between threads of control, and should the compiler re-order fields or otherwise change those data structures between the two builds, errors may result.

If the release is a patch release, do the following:

1. Shut down the old version of the application.
2. Install the new version of the application by relinking or installing a new version of the Berkeley DB shared library.
3. Restart the application.

Otherwise, if the application **does not** have a Berkeley DB transactional environment, the application may be installed in the field using the following steps:

1. Shut down the old version of the application.
2. Remove any Berkeley DB environment using the `DB_ENV->remove()` method or an appropriate system utility.
3. Recompile and install the new version of the application.
4. If necessary, upgrade the application's databases. See Database upgrade for more information.
5. Restart the application.

Otherwise, if the application has a Berkeley DB transactional environment, but neither the log file nor database formats need upgrading, the application may be installed in the field using the following steps:

1. Shut down the old version of the application.
2. Run recovery on the database environment using the `DB_ENV->open()` method or the `db_recover` utility.
3. Remove any Berkeley DB environment using the `DB_ENV->remove()` method or an appropriate system utility.
4. Recompile and install the new version of the application.
5. Restart the application.

If the application has a Berkeley DB transactional environment, and the log files need upgrading but the databases do not, the application may be installed in the field using the following steps:

1. Shut down the old version of the application.
2. Still using the old version of Berkeley DB, run recovery on the database environment using the `DB_ENV->open()` method, or the `db_recover` utility.
3. If you used the `DB_ENV->open()` method to run recovery, make sure that the Berkeley DB environment is removed using the `DB_ENV->remove()` method or an appropriate system utility.
4. Archive the database environment for catastrophic recovery using the `db_archive` utility as described in the Database and log file archival section in the *Berkeley DB Programmer's Reference Guide*.
5. Recompile and install the new version of the application.
6. Force a checkpoint using the `DB_ENV->txn_checkpoint()` method or the `db_checkpoint` utility. If you use the `db_checkpoint` utility, make sure to use the new version of the utility; that is, the version that came with the release of Berkeley DB to which you are upgrading.

Note that forcing a checkpoint might result in warning messages about log files that are being skipped. This is normal, and can be safely ignored.

7. Remove unnecessary log files from the environment using the `-d` option on the `db_archive` utility, or from an application which calls the `DB_ENV->log_archive()` method with the `DB_ARCH_REMOVE` flag.

Note that removing log files in this way might result in warning messages about log files that are being skipped. This is normal, and can be safely ignored.

Note that if you are upgrading a replicated application, then you should *not* perform this step until all of the replication sites have been upgraded to the current release level. If you run this site before all your sites are upgraded, then errors can occur in your replication activities because important version information might be lost.

8. Restart the application.

Otherwise, if the application has a Berkeley DB transactional environment and the databases need upgrading, the application may be installed in the field using the following steps:

1. Shut down the old version of the application.
2. Still using the old version of Berkeley DB, run recovery on the database environment using the `DB_ENV->open()` method, or the `db_recover` utility.
3. If you used the `DB_ENV->open()` method to run recovery, make sure that the Berkeley DB environment is removed using the `DB_ENV->remove()` method or an appropriate system utility.
4. Archive the database environment for catastrophic recovery using the `db_archive` utility as described in the Database and log file archival section in the *Berkeley DB Programmer's Reference Guide*.
5. Recompile and install the new version of the application.
6. Upgrade the application's databases. See Database upgrade for more information.
7. Archive the database for catastrophic recovery again (using different media than before, of course). Note: This archival is not strictly necessary. However, if you have to perform catastrophic recovery after restarting the application, that recovery must be done based on the last archive you have made. If you make this second archive, you can use it as the basis of that catastrophic recovery. If you do not make this second archive, you have to use the archive you made in step 4 as the basis of your recovery, and you have to do a full upgrade on it before you can apply log files created after the upgrade to it.
8. Force a checkpoint using the `DB_ENV->txn_checkpoint()` method or the `db_checkpoint` utility. If you use the `db_checkpoint` utility, make sure to use the new version of the utility; that is, the version that came with the release of Berkeley DB to which you are upgrading.

Note that forcing a checkpoint might result in warning messages about log files that are being skipped. This is normal, and can be safely ignored.

9. Remove unnecessary log files from the environment using the `-d` option on the `db_archive` utility, or from an application which calls the `DB_ENV->log_archive()` method with the `DB_ARCH_REMOVE` flag.

Note that removing log files in this way might result in warning messages about log files that are being skipped. This is normal, and can be safely ignored.

Note that if you are upgrading a replicated application, then you should *not* perform this step until all of the replication sites have been upgraded to the current release level. If you run this site before all your sites are upgraded, then errors can occur in your replication activities because important version information might be lost.

10. Restart the application.

Finally, Berkeley DB supports the live upgrade of a replication group, by allowing mixed version operation (replication sites running at the newer software version can inter-operate with older version sites). All client sites must be upgraded first; the master site must be upgraded last. In other words, at all times the master must be running the lowest version of Berkeley DB. To upgrade a replication group, you must:

1. Bring all clients up to date with the master (that is, all clients must be brought up to the most current log record as measured by the master's log sequence number (LSN)).
2. Perform the upgrade procedures described previously on each of the individual database environments that are part of the replication group. Each individual client may be upgraded and restarted to join the replication group.
3. Shut down the master site and upgrade that site last.

During live replication upgrade, while sites are running at different versions, you cannot add or remove sites from the replication group. Sites can be added or removed only after the entire group is upgraded.

Also, all removal of log files must be suspended throughout this entire procedure, so that there is no chance of a client needing internal initialization.

Alternatively, it may be simpler to discard the contents of all of the client database environments, upgrade the master database environment, and then re-add all of the clients to the replication group using the standard replication procedures for new sites.

Chapter 3. Upgrading Berkeley DB 4.6 applications to Berkeley DB 4.7

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 4.6 release interfaces to the Berkeley DB 4.7 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

Run-time configuration

In historic Berkeley DB releases, there were separate sleep and yield functions to be configured at run-time using the `db_env_set_func_sleep` and `db_env_set_func_yield` functions. These functions have been merged in the Berkeley DB 4.7 release. The replacement function should always yield the processor, and optionally wait for some period of time before allowing the thread to run again.

Applications using the Berkeley DB run-time configuration interfaces should merge the functionality of their sleep and yield functions into a single configuration function.

In the 4.7 Berkeley DB release, the `db_env_set_func_map` and `db_env_set_func_unmap` functions have been replaced. This change fixes problems where applications using the Berkeley DB run-time configuration interfaces could not open multiple `DB_ENV` class handles for the same database environment in a single application or join existing database environments from within multiple processes.

Applications wanting to replace the Berkeley DB region creation functionality should replace their `db_env_set_func_map` and `db_env_set_func_unmap` calls with a call to the `db_env_set_func_region_map` function. Applications wanting to replace the Berkeley DB region file mapping functionality should replace their `db_env_set_func_map` and `db_env_set_func_unmap` calls with a call to the `db_env_set_func_file_map` function.

Replication API

The Berkeley DB base replication API `DB_ENV->rep_elect()`, `DB_ENV->rep_get_nsites()`, `DB_ENV->rep_set_nsites()`, `DB_ENV->rep_get_priority()` and `DB_ENV->rep_set_priority()` methods now take arguments of type `u_int32_t` rather than `int`. Applications may need to change the types of arguments to these methods, or cast arguments to these methods to avoid compiler warnings.

Tcl API

The Berkeley DB Tcl API does not attempt to avoid evaluating input as Tcl commands. For this reason, it may be dangerous to pass unreviewed user input through the Berkeley DB Tcl API, as the input may subsequently be evaluated as a Tcl command. To minimize the effectiveness of a Tcl injection attack, the Berkeley DB Tcl API in the 4.7 release routine resets process' effective user and group IDs to the real user and group IDs.

DB_ENV->set_intermediate_dir

Historic releases of Berkeley DB contained an undocumented DB_ENV method named DB_ENV->set_intermediate_dir, which configured the creation of any intermediate directories needed during recovery. This method has been standardized as the DB_ENV->set_intermediate_dir_mode() method.

Applications using DB_ENV->set_intermediate_dir should be modified to use the DB_ENV->set_intermediate_dir_mode() method instead.

Log configuration

In the Berkeley DB 4.7 release, the logging subsystem is configured using the DB_ENV->log_set_config() method instead of the previously used DB_ENV->set_flags() method.

The DB_ENV->set_flags() method no longer accepts the flags DB_DIRECT_LOG, DB_DSYNC_LOG, DB_LOG_INMEMORY or DB_LOG_AUTOREMOVE. Applications should be modified to use the equivalent flags accepted by the DB_ENV->log_set_config() method.

Previous DB_ENV->set_flags() flag	Replacement DB_ENV->log_set_config() flag
DB_DIRECT_LOG	DB_LOG_DIRECT
DB_DSYNC_LOG	DB_LOG_DSYNC
DB_LOG_INMEMORY	DB_LOG_IN_MEMORY
DB_LOG_AUTOREMOVE	DB_LOG_AUTO_REMOVE

Upgrade Requirements

The log file format changed in the Berkeley DB 4.7 release.

No database formats changed in the Berkeley DB 4.7 release.

The Berkeley DB 4.7 release does not support live replication upgrade from the 4.2 or 4.3 releases, only from the 4.4 and later releases.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Berkeley DB 4.7.25 Change Log

Database or Log File On-Disk Format Changes:

1. The log file format changed in 4.7.

New Features:

1. The lock manager may now be fully partitioned, improving performance on some multi-CPU systems. [#15880]

2. Replication groups are now architecture-neutral, supporting connections between differing architectures (big-endian or little-endian, independent of structure padding). [#15787] [#15840]
3. Java: A new Direct Persistence Layer adds a built-in Plain Old Java Object (POJO)-based persistent object model, which provides support for complex object models without compromises in performance. For an introduction to the Direct Persistence Layer API, see Getting Started with Data Storage. [#15936]
4. Add the DB_ENV->set_intermediate_dir_mode method to support the creation of intermediate directories needed during recovery. [#15097]
5. The DB_ENV->failchk method can now abort transactions for threads, which have failed while blocked on a concurrency lock. This significantly decreases the need for database environment recovery after thread of control failure. [#15626]
6. Replication Manager clients now can be configured to monitor the connection to the master using heartbeat messages, in order to promptly discover connection failures. [#15714]
7. The logging system may now be configured to pre-zero log files when they are created, improving performance on some systems. [#15758]

Database Environment Changes:

1. Restructure aborted page allocation handling on systems without an ftruncate system call. This enables the Berkeley DB High Availability product on systems, which do not support ftruncate. [#15602]
2. Fix a bug where closing a database handle after aborting a transaction which included a failed open of that handle could result in application failure. [#15650]
3. Fix minor memory leaks when closing a private database environment. [#15663]
4. Fix a bug leading to a panic of "unpinned page returned" if a cursor was used for a delete multiple times and deadlocked during one of the deletes. [#15944]
5. Optionally signal processes still running in the environment before running recovery. [#15984]

Concurrent Data Store Changes:

None.

General Access Method Changes:

1. Fix a bug where closing a database handle after aborting a transaction which included a failed open of that database handle could result in application failure. [#15650]
2. Fix a bug that could cause panic in a database environment configured with POSIX-style thread locking, if a database open failed. [#15662]

3. Fix bug in the DB->compact method which could cause a panic if a thread was about to release a page while another thread was truncating the database file. [#15671]
4. Fix an obscure case of interaction between a cursor scan and delete that was prematurely returning DB_NOTFOUND. [#15785]
5. Fix a bug in the DB->compact method where if read-uncommitted was configured, a reader reading uncommitted data may see an inconsistent entry between when the compact method detects an error and when it aborts the enclosing transaction. [#15856]
6. Fix a bug in the DB->compact method where a thread of control may fail if two threads are compacting the same section of a Recno database. [#15856]
7. Fix a bug in DB->compact method, avoid an assertion failure when zero pages can be freed. [#15965]
8. Fix a bug return a non-zero error when DB->truncate is called with open cursors. [#15973]
9. Fix a bug add HANDLE_DEAD checking for DB cursors. [#15990]
10. Fix a bug to now generate errors when DB_SEQUENCE->stat is called without first opening the sequence. [#15995]
11. Fix a bug to no longer dereference a pointer into a hash structure, when hash functionality is disabled. [#16095]

Btree Access Method Changes:

None.

Hash Access Method Changes:

1. Fix a bug where a database store into a Hash database could self-deadlock in a database environment configured for the Berkeley DB Concurrent Data Store product, and with a free-threaded DB_ENV or DB handle. [#15718]

Queue Access Method Changes:

1. Fix a bug that could cause a put or delete of a queue element to return a DB_NOTGRANTED error, if blocked. [#15933]

Recno Access Method Changes:

1. Expose db_env_set_func_malloc, db_env_set_func_realloc, and db_env_set_func_free through the Windows API for the DB dll. [#16045]

C-specific API Changes:

None.

Java-specific API Changes:

1. Fix a bug where enabling MVCC on a database through the Java API was ignored. [#15644]

2. Fixed memory leak bugs in error message buffering in the Java API. [#15843]
3. Fix a bug where Java SecondaryConfig was not setting SecondaryMultiKeyCreator from the underlying db handle [OTN FORUM]
4. Fix a bug so that getStartupComplete will now return a boolean instead of an int. [#16067]
5. Fix a bug in the Java API, where Berkeley DB would hang on exit when using replication. [#16142]

Direct Persistence Layer (DPL), Bindings and Collections API:

1. A new Direct Persistence Layer adds a built-in Plain Old Java Object (POJO)-based persistent object model, which provides support for complex object models without compromises in performance. For an introduction to the Direct Persistence Layer API, see Getting Started with Data Storage. [#15936]
2. Fixed a bug in the remove method of the Iterator instances returned by the StoredCollection.iterator method in the collections package. This bug caused ArrayIndexOutOfBoundsException in some cases when calling next, previous, hasNext or hasPrevious after calling remove. (Note that this issue does not apply to StoredIterator instances returned by the StoredCollection.storedIterator method.) This bug was reported in this forum thread: <https://forums.oracle.com/forums/thread.jspa?messageID=2187896> [#15858]
3. Fixed a bug in the remove method of the StoredIterator instances returned by StoredCollection.storedIterator method in the collections package. If the sequence of methods next-remove-previous was called, previous would sometimes return the removed record. If the sequence of methods previous-remove-next was called, next would sometimes return the removed record. (Note that this issue does not apply to Iterator instances returned by the StoredCollection.iterator method.) [#15909]
4. Fixed a bug that causes a memory leak for applications where many Environment objects are opened and closed and the CurrentTransaction or TransactionRunner class is used. The problem was reported in this JE Forum thread: <https://forums.oracle.com/forums/thread.jspa?messageID=1782659> [#15444]
5. Added StoredContainer.areKeyRangesAllowed method. Key ranges and the methods in SortedMap and SortedSet such as subMap and subSet are now explicitly disallowed for RECNO and QUEUE databases -- they are only supported for BTREE databases. Before, using key ranges in a RECNO or QUEUE database did not work, but was not explicitly prohibited in the Collections API. [#15936]

Tcl-specific API Changes:

1. The Berkeley DB Tcl API does not attempt to avoid evaluating input as Tcl commands. For this reason, it may be dangerous to pass unreviewed user input through the Berkeley DB Tcl API, as the input may subsequently be evaluated as a Tcl command. To minimize the effectiveness of a Tcl injection attack, the Berkeley DB Tcl API in the 4.7 release routine resets process' effective user and group IDs to the real user and group IDs. [#15597]

RPC-specific Client/Server Changes:

None.

Replication Changes:

1. Fix a bug where a master failure resulted in multiple attempts to perform a "fast election"; subsequent elections, when necessary, now use the normal nsites value. [#15099]
2. Replication performance enhancements to speed up failover. [#15490]
3. Fix a bug where replication could self-block in a database environment configured for in-memory logging. [#15503]
4. Fix a bug where replication would attempt to read log file version numbers in a database configured for in-memory logging. [#15503]
5. Fix a bug where log files were not removed during client initialization in a database configured for in-memory logging. [#15503]
6. The 4.7 release no longer supports live replication upgrade from the 4.2 or 4.3 releases, only from the 4.4 and later releases. [#15602]
7. Fix a bug where replication could re-request missing records on every arriving record. [#15629]
8. Change the DB_ENV->rep_set_request method to use time, not the number of messages, when re-requesting missed messages on a replication client. [#15629]
9. Fix a minor memory leak on the master when updating a client during internal initialization. [#15634]
10. Fix a bug where a client error when syncing with a new replication group master could result in an inability to ever re-join the group. [#15648]
11. Change dbenv->rep_set_request to use time-based values instead of counters. [#15682]
12. Fix a bug where a LOCK_NOTGRANTED error could be returned from the DB_ENV->rep_process_message method, instead of being handled internally by replication. [#15685]
13. Fix a bug where the Replication Manager would reject a fresh connection from a remote site that had crashed and restarted, displaying the message: "redundant incoming connection will be ignored". [#15731]
14. The Replication Manager now supports dynamic negotiation of the best available wire protocol version, on a per-connection basis. [#15783]
15. Fix a bug, which could lead to slow performance of internal initialization under the Replication Manager, as evidenced by "queue limit exceeded" messages in verbose replication diagnostic output. [#15788]

16. Fix a bug where replication control message were not portable between replication clients with different endian architectures. [#15793]
17. Add a configuration option to turn off Replication Manager's special handling of elections in 2-site groups. [#15873]
18. Fix a bug making it impossible to call replicationManagerAddRemoteSite in the Java API after having called replicationManagerStart. [#15875]
19. Fix a bug where the DB_EVENT_REP_STARTUPDONE event could be triggered too early. [#15887]
20. Fix a bug where the rcvd_ts timestamp is reset when the user just changes the threshold. [#15895]
21. Fix a bug where the master in a 2-site replication group might wait for client acknowledgement, even when there was no client connected. [#15927]
22. Fix a bug, clean up and restart internal init if master log is gone. [#16006]
23. Fix a bug, ignore page messages that are from an old internal init. [#16075] [#16059]
24. Fix a bug where checkpoint records do not indicate a database was a named in-memory database. [#16076]
25. Fix a bug with in-memory replication, where we returned with the log region mutex held in an error path, leading to self-deadlock. [#16088]
26. Fix a bug which causes the DB_REP_CHECKPOINT_DELAY setting in rep_set_timeout() to be interpreted in seconds, rather than microseconds. [#16153]

XA Resource Manager Changes:

1. Fix a bug where the DB_ENV->failchk method and replication in general could fail in database environments configured for XA. [#15654]

Locking Subsystem Changes:

1. Fix a bug causing a lock or transaction timeout to not be set properly after the first timeout triggers on a particular lock id. [#15847]
2. Fix a bug that would cause a trap if DB_ENV->lock_id_free was passed an invalid locker id. [#16005]
3. Fix a bug when thread tracking is enabled where an attempt is made to release a mutex that is not lock. [#16011]

Logging Subsystem Changes:

1. Fix a bug, handle zero-length log records doing HA sync with in-memory logs. [#15838]
2. Fix a bug that could cause DB_ENV->failcheck to leak log region memory. [#15925]

3. Fix a bug where the abort of a transaction that opened a database could leak log region memory. [#15953]
4. Fix a bug that could leak memory in the DB_ENV->log_archive interface if a log file was not found. [#16013]

Memory Pool Subsystem Changes:

1. Fix multiple MVCC bugs including a race, which could *result in incorrect data being returned* to the application. [#15653]
2. Fixed a bug that left an active file in the buffer pool after a database create was aborted. [#15918]
3. Fix a bug where there could be uneven distribution of pages if a single database and multiple cache regions are configured. [#16015]
4. Fix a bug where DB_MPOOLFILE->set_maxsize was dropping the wrong mutex after open. [#16050]

Mutex Subsystem Changes:

1. Fix a bug where mutex contention in database environments configured for hybrid mutex support could result in performance degradation. [#15646]
2. Set the DB_MUTEX_PROCESS_ONLY flag on all mutexes in private environments, they can't be shared and so we can use the faster, intra-process only mutex implementations [#16025]
3. Fix a bug so that mutexes are now removed from the environment signature if mutexes are disabled. [#16042]

Transaction Subsystem Changes:

1. Fix a bug that could cause a checkpoint to selfblock attempting to flush a file, when the file handle was closed by another thread during the flush. [#15692]
2. Fix a bug that could cause DB_ENV->failcheck to hang if there were pending prepared transactions in the environment. [#15925]
3. Prepared transactions will now use the sync setting from the environment. Default to flushing the log on commit (was nosync). [#15995]
4. If __txn_getactive fails, we now return with the log region mutex held. This is not a bus since __txn_getactive cannot really fail. [#16088]

Utility Changes:

1. Update db_stat with -x option for mutex stats
2. Fix an incorrect assumption about buffer size when getting an overflow page in db_verify. [#16064]

Configuration, Documentation, Sample Application, Portability and Build Changes:

1. Fix an installation bug where the Berkeley DB PHP header file was not installed in the correct place.
2. Merge the run-time configuration sleep and yield functions. [#15037]
3. Fix Handle_DEAD and other expected replication errors in the C++ sample application ReqQuoteExample.cpp. [15568]
4. Add support for monotonic timers. [#15670]
5. Fix bugs where applications using the db_env_func_map and db_env_func_unmap run-time configuration functions could not join existing database environments, or open multiple DB_ENV handles for a single environment. [#15930]
6. Add documentation about building Berkeley DB for VxWorks 6.x.
7. Remove the HAVE_FINE_GRAINED_LOCK_MANAGER flag, it is obsolete in 4.7.
8. Fix a bug in ex_rep, add a missing break which could cause a segment fault.
9. Fix build warnings from 64 bit Windows build. [#16029]
10. Fix an alignment bug on ARM Linux. Force the assignment to use memcpy. [#16125]
11. Fix a bug in the Windows specific code of ex_sequence.c, where there was an invalide printf specifier. [#16131]
12. Improve the timer in ex_tpcb to use high resolution timers. [#16154]
13. Mention in the documentation that env->open() requires DB_THREAD to be specified when using repmgr. [#16163]
14. Disable support for mmap on Windows CE. The only affect is that we do not attempt to mmap small read only databases into the mpool. [#16169]

Chapter 4. Upgrading Berkeley DB 4.5 applications to Berkeley DB 4.6

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 4.5 release interfaces to the Berkeley DB 4.6 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

C API cursor handle method names

In the Berkeley DB 4.6 release, the C API DBC handle methods have been renamed for consistency with the C++ and Java APIs. The change is the removal of the leading "c_" from the names, as follows:

DBC->c_close
Renamed DBC->close

DBC->c_count
Renamed DBC->count

DBC->c_del
Renamed DBC->del

DBC->c_dup
Renamed DBC->dup

DBC->c_get
Renamed DBC->get

DBC->c_pget
Renamed DBC->pget

DBC->c_put
Renamed DBC->put

The old DBC method names are deprecated but will continue to work for some number of future releases.

DB_MPOOLFILE->put

The DB_MPOOLFILE->put() method takes a new parameter in the Berkeley DB 4.6 release, a page priority. This parameter allows applications to specify the page's priority when returning the page to the cache.

Applications calling the DB_MPOOLFILE->put() method can upgrade by adding a DB_PRIORITY_UNCHANGED parameter to their calls to the DB_MPOOLFILE->put() method. This will result in no change in the application's behavior.

B_MPOOLFILE->set

The DB_MPOOLFILE->set method has been removed from the Berkeley DB 4.6 release. Applications calling this method can upgrade by removing all calls to the method. This will result in no change in the application's behavior.

Replication Events

It is now guaranteed the `DB_EVENT_REP_STARTUPDONE` event will be presented to the application after the corresponding `DB_EVENT_REP_NEWMASTER` event, even in the face of extreme thread-scheduling anomalies. (In previous releases, if the thread processing the `NEWMASTER` message was starved, and `STARTUPDONE` occurred soon after, the order might have been reversed.)

In addition, the `DB_EVENT_REP_NEWMASTER` event is now presented to all types of replication applications: users of either the Replication Framework or the Base Replication API. In both cases, the `DB_EVENT_REP_NEWMASTER` event always means that a site other than the local environment has become master.

The `envid` parameter to `DB_ENV->rep_process_message()` has been changed to be of type "int" rather than "int **", and the environment ID of a new master is presented to the application along with the `DB_EVENT_REP_NEWMASTER` event. Replication applications should be modified to use the `DB_EVENT_REP_NEWMASTER` event to determine the ID of the new master.

The `envid` parameter has been removed from the `DB_ENV->rep_elect()` method and a new event type has been added. The `DB_EVENT_REP_ELECTED` event is presented to the application at the site which wins an election. In the Berkeley DB 4.6 release, the normal result of a successful election is either the `DB_EVENT_REP_NEWMASTER` event (with the winner's environment ID), or the `DB_EVENT_REP_ELECTED` event. Only one of the two events will ever be delivered.

The `DB_REP_NEWMASTER` return code has been removed from the `DB_ENV->rep_process_message()` method. Replication applications should be modified to use the `DB_EVENT_REP_NEWMASTER` and `DB_EVENT_REP_ELECTED` events to determine the existence of a new master.

DB_REP_FULL_ELECTION

The `DB_REP_FULL_ELECTION` flag historically specified to the `DB_ENV->repmgr_start()` method has been removed from the 4.6 release.

In the Berkeley DB 4.6 release, a simpler and more flexible implementation of this functionality is available. Applications needing to configure the first election of a replication group differently from subsequent elections should use the `DB_REP_FULL_ELECTION_TIMEOUT` flag to the `DB_ENV->rep_set_timeout()` method to specify a different timeout for the first election.

Verbose Output

When an error occurs in the Berkeley DB library, an exception is thrown or an error return value is returned by the interface. In some cases, however, the exception or returned value may be insufficient to completely describe the cause of the error, especially during initial application debugging. Applications can configure Berkeley DB for verbose messages to be output when an error occurs, but it's a common cause of confusion for new users that no verbose messages are available by default.

In the Berkeley DB 4.6 release, verbose messages are configured by default. For the C and C++ APIs, this means the default configuration when applications first create DB or DB_ENV handles is as if the DB_ENV->set_errfile() or DB->set_errfile() methods were called with the standard error output (stderr) specified as the FILE * argument. Applications wanting no output at all can turn off this default configuration by calling the DB_ENV->set_errfile() or DB->set_errfile() methods with NULL as the FILE * argument. Additionally, explicitly configuring the error output channel using any of the DB_ENV->set_errfile(), DB_ENV->set_errcall(), DbEnv::set_error_stream() or Db::set_error_stream() methods will also turn off this default output for the application.

Applications which configure Berkeley DB with any error output channel should not require any changes.

Applications which depend on having no output from the Berkeley DB library by default, should be changed to call the DB_ENV->set_errfile() or DB->set_errfile() methods with NULL as the FILE * argument.

DB_VERB_REPLICATION

The DB_VERB_REPLICATION flag no longer requires the Berkeley DB library be built with the --enable-diagnostic configuration option to output additional replication logging information.

Windows 9X

Berkeley DB no longer supports process-shared database environments on Windows 9X platforms; the DB_PRIVATE flag must always be specified to the DB_ENV->open() method.

Upgrade Requirements

The log file format changed in the Berkeley DB 4.6 release.

The format of Hash database pages was changed in the Berkeley DB 4.6 release, and items are now stored in sorted order. **The format changes are entirely backward-compatible, and no database upgrades are needed.** However, upgrading existing databases can offer significant performance improvements. Note that databases created using the 4.6 release may not be usable with earlier Berkeley DB releases.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Berkeley DB 4.6.21 Change Log

4.6.21 Patches:

1. Fix a bug where mutex contention in database environments configured for hybrid mutex support could result in performance degradation. [#15646]
2. Fix a bug where closing a database handle after aborting a transaction which included a failed open of that database handle could result in application failure. [#15650]

3. Fix multiple MVCC bugs including a race which *could result in incorrect data being returned* to the application. [#15653]
4. Fix a bug where a database store into a Hash database could self-deadlock in a database environment configured for the Berkeley DB Concurrent Data Store product and with a free-threaded DB_ENV or DB handle. [#15718]
5. Fix an installation bug where Berkeley DB's PHP header file was not installed in the correct place.

4.6.19 Patches

1. Fix a bug where a client in a two-site replication group could become master, after failure of the existing master, even if the client had priority 0. [#15388]
2. Fix a bug where 32-bit builds on 64-bit machines could immediately core dump because of a misaligned access. [#15643]
3. Fix a bug where attempts to configure a database for MVCC in the Java API were silently ignored. [#15644]
4. Fix a bug where database environments configured for replication and verbose output could drop core. [#15651]

Database or Log File On-Disk Format Changes:

1. The on-disk log format has changed.
2. The format of Hash database pages was changed in the Berkeley DB 4.6 release, and items are now stored in sorted order. *The format changes are entirely backward-compatible, and no database upgrades are needed.* However, upgrading existing databases can offer significant performance improvements. Note that databases created using the 4.6 release may not be usable with earlier Berkeley DB releases.

New Features:

1. Add support for a cursor DB_PREV_DUP flag, which moves the cursor to the previous key/data pair if it's a duplicate of the current key/data pair. [#4801]
2. Add the ability to set cache page priority on a database or cursor handle. [#11886]
3. Add verbose output tracing for filesystem operations. [#13760]
4. Port Berkeley DB to Qualcomm's Binary Runtime Environment for Wireless (BREW). [#14562]
5. Port Berkeley DB to WinCE. [#15312]
6. Port Berkeley DB to S60. [#15371]
7. Add a key_exists method to the DB handle. [#15374]

8. Applications may now begin processing new transactions while previously prepared, but unresolved, transactions are still pending. [#14754]
9. Significant performance improvements in the Hash access method. [#15017]

Database Environment Changes:

1. Add support to close open file handles in the case of catastrophic database environment failure so applications that do not exit and restart on failure won't leak file handles. [#6538]
2. Replace the Berkeley DB shared memory allocator with a new implementation, intended to decrease the performance drop-off seen in database environments having working sets that are larger than the cache, especially database environments with multiple cache page sizes. [#13122]
3. Fix a bug that would incorrectly cause a thread to appear to be in the Berkeley DB API after a call to db_create. [#14562]
4. Allow database close prior to resolving all transactions updating the database. [#14785]
5. Fix a bug where the db_stat utility -Z flag and the statistics method's DB_STAT_CLEAR flag could clear mutex statistics too quickly, leading to incorrect values being displayed. [#15032]
6. Fix a bug where removal of a file after an open/close pair spanning the most recent checkpoint log-sequence-numbers made recovery fail. [#15092]
7. Fix a bug that could leave an environment unrecoverable if FTRUNCATE was not set and a roll-forward to a timestamp was interrupted between the truncation of the log and the recording of aborted allocations. [#15108]
8. Fix a bug where recovery of a rename operation could fail if the rename occurred in a directory that no longer existed. [#15119]
9. Fix a bug that could cause recovery to report a "File exists" error if a committed create was partially recovered by a previously failed recovery operation. [#15151]
10. Fix a bug where the DbEnv.get_thread_count method implementation was missing from the Berkeley DB 4.5 release. [#15201]
11. Fix a bug where replication operations were not reported properly when the DbEnv.failchk method was called. [#15094]
12. Fixed a bug that caused SEQ->remove not to use a transaction if the sequence was opened on a transactional database handle but no transaction was specified on the call. [#15235]
13. Fix a bug where accesses to the database environment reference count could race, causing the DB_ENV->remove method to incorrectly remove or not remove a database environment. [#15240]

14. Fix a bug that could cause a recovery failure if a partial record was written near the end of a log file before a crash and then never overwritten after recovery runs and before a log file switch occurs. [#15302]
15. Fix a bug that could fire a diagnostic assertion if an error occurred during a database environment open. [#15309]
16. Fix a bug where memp_trickle attempts to flush an infinite number of buffers. [#15342]
17. Cause application updates of the DB_ENV->set_mp_max_write values to affect already running cache flush operations. [#15342]
18. Fix a bug which could cause system hang if a checkpoint happened at the same time as a database file create or rename. [#15346]
19. Fix a bug which could cause application failure if the open of a subdatabase failed while other database opens were happening. [#15346]
20. Fix a bug that could cause recovery to not process a transaction properly if the transaction was started before the transaction IDs were reset but did not put its first record into the log until after the txn_recycle record. [#15400]
21. Fix a bug that could cause a thread in cache allocation to loop infinitely. [#15406]
22. Fix a bug that could cause recovery to report a Log Sequence Error on systems without the ftruncate system call where a page allocation occurred and the database metadata page was forced out of cache without being marked dirty and then had to be recovered. [#15441]
23. Fix a bug on systems lacking the ftruncate system call, where a page may be improperly linked into the free list if archive recovery was done in multiple steps, that is, applying additional logs to the same databases. [#15557]

Concurrent Data Store Changes:

None.

General Access Method Changes:

1. Add a feature where applications can specify a custom comparison function for the Hash access method [#4109]
2. Open, create, close and removal of non-transactional databases is are longer logged in transactional database environments unless debug logging is enabled. [#8037]
3. Add the ability to set cache page priority on a database or cursor handle. [#11886]
4. fix a bug where the DB_ENV->fileid_reset method failed when called on on encrypted or check-summed databases. [#13990]
5. Fix a bug where the DB->fd method failed when called on in-memory databases. [#14157]

6. Fix a bug where an attempt to open a Recno database with a backing file that does not exist could report an error because it couldn't remove a temporary file. [#14160]
7. Reverse a change found in previous releases which disallowed setting "partial" flags on key DBTs for DB and DbCursor put method calls. [#14520]
8. Fix a bug where transactional file operations, such as remove or rename, could leak file handles. [#15222]
9. Fix a bug that could cause the in-memory sorted freelist used by the DB->compact method not to be freed if transaction or lock timeouts were set in the environment. [#15292]
10. Add the DB->get_multiple method, which returns if the DB handle references a "master" database in the physical file. [#15352]
11. Fix a bug that could cause an DB_INORDER, DB->get method DB_CONSUME operation to loop if the Queue database was missing a record due to a rollback by a writer or a non-queue insert in the queue. [#15452]
12. Fix a bug preventing database removal after application or system failure in a database environment configured for in-memory logging. [#15459]

Btree Access Method Changes:

None.

Hash Access Method Changes:

1. Change the internal format of Hash database pages, storing items in sorted order. There are no externally visible changes, and hash databases using historic on-page formats do not require an explicit upgrade. (However, upgrading existing databases can offer significant performance improvements.) [#15017]
2. Fix a bug preventing LSNs from being reset on hash databases when the databases were configured with a non-standard hash function. [#15567]

Queue Access Method Changes:

1. Fix a bug which could cause a Queue extent file to be incorrectly removed if an empty extent file was being closed by one thread and being updated by another thread (which was using random access operations). [#9101]

Recno Access Method Changes:

None.

C++-specific API Changes:

None.

Java-specific API Changes:

1. Add a feature where an exception is thrown by the Java API, the Berkeley DB error message is now included in the exception object. [#11870]

2. Fix a bug which can cause a JVM crash when doing a partial get operation. [#15143]
3. Fix a bug which prevented the use of Berkeley DB sequences from Java. [#15220]
4. Fix multiple bugs where DBTs were not being copied correctly in the Java replication APIs. [#15223]
5. Add transaction.commitWriteNoSync to the Java API. [#15376]

Java collections and bind API Changes:

1. Change SerialBinding to use the current thread's context class loader when loading application classes. This allows the JE jar file to be deployed in application servers and other containers as a shared library rather than as an application jar. [#15447]
2. Tuple bindings now support the java.math.BigInteger type. Like other tuple binding values, BigInteger values are sorted in natural integer order by default, without using a custom comparator. For details please see the Javadoc for: com.sleepycat.bind.tuple.TupleInput.readBigInteger
com.sleepycat.bind.tuple.TupleOutput.writeBigInteger
com.sleepycat.bind.tuple.BigIntegerBinding [#15244]
3. Add tuple binding methods for reading and writing packed int and long values. Packed integer values take less space, but take slightly more processing time to read and write. See: TupleInput.readPackedInt TupleInput.getPackedIntByteLength
TupleInput.readPackedLong TupleInput.getPackedLongByteLength
TupleOutput.writePackedInt TupleOutput.writePackedLong PackedInteger [#15422]
4. The Collections API has been enhanced so that auto-commit works for the standard Java Iterator.remove(), set() and add() methods. Previously it was necessary to explicitly begin and commit a transaction in order to call these methods, when the underlying Database was transactional. Note that starting a transaction is still necessary when calling these methods if the StoredCollection.storedIterator method is used. [#15401]
5. Fix a bug that causes a memory leak for applications where both of the following are true: many Environment objects are opened and closed, and the CurrentTransaction or TransactionRunner class is used. [#15444]

Tcl-specific API Changes:

None.

RPC-specific Client/Server Changes:

None.

Replication Changes:

1. Fix a bug where transactions could be rolled-back if an existing replication group master was partitioned and unable to participate in an election. [#14752]
2. Add a new event when a replication manager framework master fails to send and confirm receipt by clients of a "permanent" message. [#14775]

3. Fix a race where multiple threads might attempt to process a LOGREADY condition. [#14902]
4. Change the DB_VERB_REPLICATION flag to no longer require the Berkeley DB library be built with the --enable-diagnostic configuration option to output additional replication logging information. [#14991]
5. Fix a bug with elections occurring during internal init of a replication client site. [#15057]
6. Fix lockout code to lockout message threads and API separately. Send indication that log requests is for internal init. [#15067]
7. Replication manager changed to retry host-name look-up failures, since they could be caused by transient name server outage. [#15081]
8. Fix a bug which led to memory corruption when the sending of a bulk buffer resulted in an error. [#15100]
9. A throttling limit of 10 megabytes is now set by default in a newly created database environment (see the DbEnv.rep_set_limit method). [#15115]
10. Fix a bug in ALL_REQ handling where master could get a DB_NOTFOUND. [#15116]
11. Fix a bug which could lead to client sites repeatedly but unproductively calling for an election, when a master site already exists. [#15128]
12. Modify gap processing algorithms so XXX_MORE messages ask for data beyond what it just processed, not an earlier gap that might exist. [#15136]
13. Fixed a bug in the ex_rep example application which could cause the last few transactions to disappear when shutting down the sites of the replication group gracefully. [#15162]
14. Fix a bug where if a client crashed during internal init, its database environment would be left in a confused state, making it impossible to synchronize again with the master. [#15177]
15. Fix a bug where election flags are not cleared atomically with the setting of the new master ID. [#15186]
16. Fix a bug which would cause Berkeley DB to crash if an internal init happened when there were no database files at the master. [#15227]
17. It is now guaranteed that the DB_EVENT_REP_STARTUPDONE event will be presented to the application after the corresponding DB_EVENT_REP_NEWMASTER event, even in the face of extreme scheduling anomalies. [#15265]
18. Fix minor memory leaks in the replication manager. [#15239] [#15256]
19. Fix a bug which caused the replication manager to lose track of a failed connection, resulting in the inability to accept a replacement connection. [#15311]
20. Fix a bug where a client starting an election when the rest of the replication group already had an established master could confuse replication management at the other

client sites, leading to failure to properly acknowledge PERM transactions from the master. [#15428]

21. Add support for reporting Replication Manager statistics. [#15430]
22. Fix a bug where a send failure during processing of a request message from a client could erroneously appear to the application as an EPERM system error. [#15436]
23. Client now sets STARTUPDONE at the end of the synchronization phase when it has caught up to the end of the master's transaction log, without requiring ongoing transactions at the master. [#15542]
24. Fix a bug in sleep-time calculation which could cause a Replication Manager failure. [#15552]

XA Resource Manager Changes:

None.

Locking Subsystem Changes:

1. Change the DB_ENV->lock_detect method to return the number of transactions timed out in addition to those were rejected due to deadlock. [#15281]

Logging Subsystem Changes:

None.

Memory Pool Subsystem Changes:

1. Fix a bug that could cause a checkpoint to hang if a database was closed while the checkpoint was forcing that file to disk and all the pages for that database were replaced in the cache. [#15135]
2. Fix a bug where a system error in closing a file could result in a core dump. [#15137]
3. Fix MVCC statistics counts for private database environments. [#15218]

Transaction Subsystem Changes:

1. Fix a bug where creating a database with the DB_TXN_NOTDURABLE flag set would still write a log record. [#15386]
2. Change transaction checkpoint to wait only for pages being updated during the checkpoint. [#14710]

Utility Changes:

1. Fix a bug that prevented db_load from handling subdatabase names that were of zero length. [#8204]
2. Fix a bug where the db_hotbackup utility did not clean out and record the log file numbers in the backup directory when both the -u and -D flags were specified. [#15395]

Configuration, Documentation, Portability and Build Changes:

1. Berkeley DB no longer supports process-shared database environments on Windows 9X platforms; the DB_PRIVATE flag must always be specified to the DB_ENV->open method. [#13766]
2. Port Berkeley DB to Qualcomm's Binary Runtime Environment for Wireless (BREW). [#14562]
3. Compile SWIG-generated code with the -fno-strict-aliasing flag when using the GNU gcc compiler. [#14953]
4. Changed include files so ENOENT is resolved on Windows. [#15078]
5. Port Berkeley DB to WinCE. [#15312]
6. Port Berkeley DB to S60. [#15371]
7. Add the db_hotbackup executable to the Windows MSI installer. [#15372]
8. Change the db_hotbackup utility to use the Berkeley DB library portability layer. [#15415]
9. Re-write the GNU gcc mutex implementation on the x86 platform to avoid compiler errors. [#15461]
10. Fix a bug with non-HFS filesystems under OS X which could affect data durability. [#15501]

Chapter 5. Upgrading Berkeley DB 4.4 applications to Berkeley DB 4.5

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 4.4 release interfaces to the Berkeley DB 4.5 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

deprecated interfaces

Some previously deprecated interfaces were removed from the Berkeley DB 4.5 release:

- The `DB_ENV->set_lk_max` method was removed. This method has been deprecated and undocumented since the Berkeley DB 4.0 release.
- The `DB->stat` method flags `DB_CACHED_COUNT` and `DB_RECORDCOUNT` were removed. These flags have been deprecated and undocumented since the Berkeley DB 4.1 release.
- The `-w` option to the `db_deadlock` utility was removed. This option has been deprecated and undocumented since the Berkeley DB 4.0 release.

DB->set_isalive

In previous releases, the function specified to the `DB_ENV->set_isalive()` method did not take a flags parameter. In the Berkeley DB 4.5 release, an additional flags argument has been added: `DB_MUTEX_PROCESS_ONLY`.

Applications configuring an is-alive function should add a flags argument to the function, and change the function to ignore any thread ID and return the status of just the process, when the `DB_MUTEX_PROCESS_ONLY` flag is specified.

DB_ENV->rep_elect

Two of the historic arguments for the `DB_ENV->rep_elect()` method have been moved from the interface to separate methods in order to make them available within the new replication manager framework.

The **priority** parameter should now be explicitly set using the `DB_ENV->rep_set_priority()` method. To upgrade existing replication applications to the Berkeley DB 4.5 `DB_ENV->rep_elect()` interface, it may be simplest to insert a call to `DB_ENV->rep_set_priority()` immediately before the existing call to `DB_ENV->rep_elect()`. Alternatively, it may make more sense to add a single call to `DB_ENV->rep_set_priority()` during database environment configuration.

The **timeout** parameter should now be explicitly set using the `DB_ENV->rep_set_timeout()` method. To upgrade existing replication applications to the Berkeley DB 4.5 `DB_ENV->rep_elect()` interface, it may be simplest to insert a call to `DB_ENV->rep_set_timeout()` immediately before the existing call to `DB_ENV->rep_elect()`. Alternatively, it may make

more sense to add a single call to `DB_ENV->rep_set_timeout()` during database environment configuration.

Replication method naming

The method names `DB_ENV->set_rep_limit`, `DB_ENV->get_rep_limit` and `DB_ENV->set_rep_transport` have been changed to `DB_ENV->rep_set_limit()`, `DB_ENV->rep_get_limit()` and `DB_ENV->rep_set_transport()` in order to be consistent with the other replication method names. That is, the characters "set_rep" and "get_rep" have been changed to "rep_set" and "rep_get".

Applications should modify the method names, no other change is required.

Replication events

One of the informational returns from the `DB_ENV->rep_process_message()` method found in previous releases of Berkeley DB has been changed to an event. The `DB_REP_STARTUPDONE` return from `DB_ENV->rep_process_message()` is now the `DB_EVENT_REP_STARTUPDONE` value to the `DB_ENV->set_event_notify()` callback.

Applications should update their handling of this event as necessary.

Memory Pool API

As part of implementing support for multi-version concurrency control, the `DB_MPOOL_DIRTY` flag is now specified to the `DB_MPOOLFILE->get()` instead of `DB_MPOOLFILE->put()`, and the `DB_MPOOLFILE->set` method has been removed. In addition, a new transaction handle parameter has been added to the `DB_MPOOLFILE->get()` method.

The `DB_MPOOL_CLEAN` flag is no longer supported.

Applications which use the memory pool API directly should update to the new API in order to use 4.5.

DB_ENV->set_paniccall

In previous Berkeley DB releases, the `DB_ENV->set_paniccall` and `DB->set_paniccall` methods were used to register a callback function, called if the database environment failed. In the 4.5 release, this functionality has been replaced by a general-purpose event notification callback function, set with the `DB_ENV->set_event_notify()` method. Applications should be updated to replace `DB_ENV->set_paniccall` and `DB->set_paniccall` calls with a call to `DB_ENV->set_event_notify()`. This also requires the callback function itself change, as the callback signatures are different.

The `DB_ENV->set_paniccall` and `DB->set_paniccall` calls are expected to be removed in a future release of Berkeley DB.

DB->set_pagesize

In previous releases, when creating a new database in a physical file which already contained databases, it was an error to specify a page size different from the existing databases in the

file. In the Berkeley DB 4.5 release, any page size specified is ignored if the file in which the database is being created already exists.

Collections API

The changes to the Collections API are compatible with prior releases, with one exception: the `Iterator` object returned by the `StoredCollection.iterator()` method can no longer be explicitly cast to `StoredIterator` because a different implementation class is now used for iterators. If you depend on the `StoredIterator` class, you must now call `StoredCollection.storedIterator()` instead. Note the `StoredIterator.close(Iterator)` static method is compatible with the new iterator implementation, so no changes are necessary if you are using that method to close iterators.

--enable-pthread_self

In previous releases, the `--enable-pthread_self` configuration option was used to force Berkeley DB to use the POSIX pthread `pthread_self` function to identify threads of control (even when Berkeley DB was configured for test-and-set mutexes). In the 4.5 release, the `--enable-pthread_self` option has been replaced with the `--enable-pthread_api` option. This option has the same effect as the previous option, but configures the Berkeley DB build for a POSIX pthread application in other ways (for example, configuring Berkeley DB to use the `pthread_self` function).

Recno backing text source files

In previous releases of Berkeley DB, Recno access method backing source text files were opened using the ANSI C `fopen` function with the "r" and "w" modes. This caused Windows systems to translate carriage-return and linefeed characters on input and output and could lead to database corruption.

In the current release, Berkeley DB opens the backing source text files using the "rb" and "wb" modes, consequently carriage-return and linefeed characters will not be translated on Windows systems.

Applications using the backing source text file feature on systems where the "r/w" and "rb/wb" modes differ should evaluate their application as part of upgrading to the 4.5 release. There is the possibility that characters have been translated or stripped and the backing source file has been corrupted. (Applications on other systems, for example, POSIX-like systems, should not require any changes related to this issue.)

Application-specific logging

In previous releases of Berkeley DB, "BEGIN" lines in the XXX.src files used to build application-specific logging support only required a log record number. In the 4.5 release, those lines require a Berkeley DB library version as well. For example, the entry:

```
BEGIN mkdir 10000
```

must now be:

```
BEGIN mkdir 44 10000
```

that is, the version of the Berkeley DB release where the log record was introduced must be included. The version is the major and minor numbers for the Berkeley DB library, with all punctuation removed. For example, Berkeley DB version 4.2 should be 42, version 4.5 should be 45.

Upgrade Requirements

The log file format changed in the Berkeley DB 4.5 release. No database formats changed in the Berkeley DB 4.5 release.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Berkeley DB 4.5.20 Change Log

Database or Log File On-Disk Format Changes:

1. The on-disk log format has changed.

New Features:

1. Multi-Version Concurrency Control for the Btree/Recno access methods.
2. A new replication framework with a default TCP/IP setup.
3. Online replication upgrades for high availability replicated 24/7 systems.
4. A new event-style notification.
5. Several enhancements to the Java Collections API including the implementation of the `size()` method.

Database Environment Changes:

1. Update the `DB_ENV->failchk` method to garbage collect per-process mutexes stranded after unexpected process failure. [#13964]
2. Fix a bug that could cause memory used to track threads for `DB_ENV->failchk` to not be reused when a thread no longer exists. [#14425]
3. Add `set_event_notify` behavior as part of new event notification in Berkeley DB. [#14534]
4. Fix a bug so that we no longer panic on `DB_ENV->close()` if a previous environment close failed to log. This condition will now return an error. [#14693]
5. Created `os_getenv`, removed `clib/getenv`, implemented Windows specific behavior. [#14942]
6. Fix a bug where it was possible to corrupt the `DB_REGISTER` information file, making it impossible for Berkeley DB applications to join database environments. [#14998]

Concurrent Data Store Changes:

1. Fix a bug where renaming a subdatabase in a Concurrent Data Store environment could fail. [#14185]

General Access Method Changes:

1. Fix a bug that could leave extra unallocated pages at the end of a database file. [#14031]
2. Optimize secondary updates when overwriting primary records. [#14075]
3. Fix a bug to prevent a trap when creating a named in-memory database and there are already temporary files open. [#14133]
4. Fix a bug which caused a trap if the key parameter to DBC->c_get was omitted with DB_CURRENT. [#14143]
5. Fix a bug with secondary cursors when the secondary has off-page duplicates. This bug resulted in incorrect primary data being returned. [#14240]
6. Improve performance when removing a subdatabase by not locking every page. [#14366]
7. Fix a bug that would not properly upgrade database files from releases 3.2.9 (and earlier) to releases 4.0 (and greater). [#14461]
8. Fix a bug that could cause a DB_READ_UNCOMMITTED get through a secondary index to return DB_SECONDARY_CORRUPT. [#14487]
9. Fix a bug so that non-transactional cursor updates of a transactional database will generate an error. [#14519]
10. Add a message when the system panics due to a page in the wrong state at its time of allocation. [#14527]
11. Fix a remove failure when attempting to remove a file that is open in another thread of control. [#14780]
12. Fix a bug where the key was not ignored when doing a cursor put with the DB_CURRENT flag. [#14988]

Btree Access Method Changes:

- a. When deleting a page don't check the next key in the parent if we are going to delete the parent too.
 - b. Need to check that the tree has not collapsed between dropping a read lock and getting the write lock. If it has collapsed we will fetch the root of the tree.
 - c. Fix a case where we fail to lock the next page before reading it.
1. Changed the implementation of internal nodes in btrees so that they no longer share references to overflow pages with leaf nodes. [#10717]

2. Fix a bug that could cause a diagnostic assertion by setting the deleted bit on a record in an internal node. [#13944]
3. Fix three problems in BTREE compaction: [#14238]
4. Fix a bug that could cause the compaction of a Btree with sorted duplicates to fail when attempting to compact an off page duplicate tree if a key could not fit in an internal node. [#14771]
5. Fix a bug that causes a loop if an empty Btree was compacted. [#14493]

Hash Access Method Changes:

1. Fix a bug to allow creation of hash pages during truncate recovery. [#14247]

Queue Access Method Changes:

1. Fix a bug where reads of data items outside the range of the queue were not kept locked to the end of the transaction, breaking serializability. [#13719]
2. Fix a bug that could cause corruption in queue extent files if multiple processes tried to open the same extent at the same time. [#14438]
3. Improve concurrency for in-place updates in queue databases. [#14918]

Recno Access Method Changes:

None.

C++-specific API Changes:

1. C++ applications that check the error code in exceptions should note that DbMemoryException has been changed to have the error code DB_BUFFER_SMALL rather than ENOMEM, to match the error returned by the C API. DbMemoryException will be thrown when a Dbt is too small to contain data returned by Berkeley DB. When a call to malloc fails, or some other resource is exhausted, a plain DbException will be thrown with error code set to ENOMEM. [#13939]

Java-specific API Changes:

1. Database.verify may now be called. This method is now static and takes a DatabaseConfig parameter. [#13971]
2. Add DB_ENV->{fileid_reset, lsn_reset} to the public API. [#14076]

Java collections and bind API Changes:

1. The com.sleepycat.collections package is now fully compatible with the Java Collections framework. [#14732]

Tcl-specific API Changes:

1. Fix a conflicting variable, sysscript.tcl. [#15051]

RPC-specific Client/Server Changes:

None.

Replication Changes:

1. Fix a bug when running with `DEBUG_ROP` or `DEBUG_WOP`. [#13394]
2. Add live replication upgrade support [#13670]
3. Fix a bug so that client databases are removed at the start of internal initialization. [#14147]
4. Fix a bug in replication internal initialization so that `data_dir` will be handled correctly. Make internal initialization resilient to multiple `data_dir` calls with the same directory. [#14489]
5. Fix a bug in the 4.2 sync-up algorithm that could result in no open files. [#14552]
6. Fix a bug when clients decide to re-request. [#14642]
7. Fix a bug where a `PERM` bulk buffer could have a zero LSN passed to the application callback. [#14675]
8. Change names of some existing replication API methods as described in [Replication method naming \(page 27\)](#). [#14723]
9. Fix a bug which could cause an election to succeed only after waiting for the timeout to expire, even when all sites responded in a timely manner. The bug was most easily visible in an election between 2 sites. [#14752]
10. Fix a bug where a process could have an old file handle to a log file. [#14797]
11. Fix a bug where a "log_more" message could be on a log file boundary. [#15034]
12. Fix a bug that could cause log corruption if a database open operation were attempted during a call to `rep_start` in another thread. [#15035]
13. Fix a bug during elections where a `vote2` arrives before its `vote1`. [#15055]
14. Fix a bug to make sure we are a client if sending a `REP_REREQUEST`. [#15066]

XA Resource Manager Changes:

None.

Locking Subsystem Changes:

1. Fix a bug that could cause a write to hang if `DB_READ_UNCOMMITTED` is enabled and it tries to reacquire a write lock. [#14919]

Logging Subsystem Changes:

1. Fix a bug so that log headers are now included in the checksum. This avoids a possible race in doing hot backups. [#11636].

2. Add a check so that some log sequence errors are diagnosed at run time rather than during recovery. [#13231]
3. Fix a bug where recovery fails if there is no disk space for the forced checkpoint that occurs at the end of processing the log. [#13986]
4. Fix a bug which could cause a page to be missing from the end of a database file if the page at the end of the file was freed while it contained data and the system was restarted before the log record for that free was flushed to disk. [#14090]
5. Fix a bug that could cause log files to be incorrectly removed by log_archive if it was run immediately after recovery. [#14874]

Memory Pool Subsystem Changes:

1. Fix a bug that could cause corruption to the buffer pool cache if a race condition was hit while using DB->compact. [#14360]
2. Fix a bug where cache pages could be leaked in applications creating temporary files for which the DB_MPOOL_NOFILE flag was set. [#14544]

Transaction Subsystem Changes:

1. Fix a bug that could cause extra empty pages to appear in a database file after recovery. [#11118]
2. Fix a bug triggered when running recovery with a feedback function that could cause a NULL pointer dereference. [#13834]
3. Fix a bug where running recovery could create duplicate entries in the data directory list. [#13884]
4. Fix a bug to not trade locks if a write lock is already owned. [#13917]
5. Fix a bug that could cause traps or hangs if the DB_TXN->set_name function is used in a multithreaded application. [#14033]
6. Fix a bug so that a transaction can no longer be committed after it had deadlocked. [#14037]
7. Fix a bug that could cause a trap during recovery if multiple operations that could remove the same extent are recovered. [#14061]
8. Fix a bug that could cause an extent file to be deleted after the last record in the extent was consumed but the consuming transaction was aborted. [#14179]
9. Fix a bug where the parent database would not use DB_READ_UNCOMMITTED in certain cases when calling DBC->c_pget. [#14361]
10. Fix a bug so that it is no longer possible to do a non-transactional cursor update on a database that is opened transactionally. [#14519]

11. Fix a bug that causes a sequence to ignore the DB_AUTO_COMMIT settings. [#14582]
12. Fix a bug, change txn_recover so that multiple processes will recover prepared transactions without requiring that the first process stay active. [#14707]
13. Fix a bug that could cause the wrong record to be deleted if a transaction had a cursor on a record with a pending delete and then replaced a record that contained overflow data or replaced a record with overflow data and that replace failed. [#14834]

Utility Changes:

1. Fix a bug that caused db_verify to not check the order on leaf pages which were the leftmost children of an internal node. [#13004]
2. Fix a bug that caused db_hotbackup to not backup queue extent files. [#13848]
3. Fix a bug so that db_verify no longer reports that an unused hash page is not fully zeroed. [#14030]
4. Fix a bug where db_stat ignored the -f option to return "fast statistics". [#14283]
5. Fix a bug that prevented the db_stat utility from opening database files with write permission so that meta data statistics would be updated. [#14755]
6. Fix a bug in db_hotbackup related to windows. Sub-directories are now ignored. [#14757]

Configuration, Documentation, Portability and Build Changes:

1. The Berkeley DB 4.3 and 4.4 releases disallowed using the --with-uniquename configuration option with the C++, Java, or RPC --enable-XXX options. The 4.5 release returns to the 4.2 release behavior, allowing those combinations of configuration options. [#14067]
2. Fix build issues when CONFIG_TEST is not enabled for Tcl. [#14507]
3. There are updated build instructions for Berkeley DB PHP module on Linux. [#14249]
4. Use libtool's "standard" environment variable names so that you can set "AR" to "ar -X64" for example, and modify both libtool and the Makefile commands. Remove the install-strip target from the Makefile, it is no longer used. [#14726]
5. Fix a bug where, when a database is opened with the DB_THREAD flag (the default in Java), and an operation in one thread causes the database to be truncated (typically when the last page in the database is freed) concurrently with a read or write in another thread, there can be arbitrary data loss, as Windows zeros out pages from the read/write location to the end of the file. [#15063]

Chapter 6. Upgrading Berkeley DB 4.3 applications to Berkeley DB 4.4

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 4.3 release interfaces to the Berkeley DB 4.4 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

DB_AUTO_COMMIT

In previous Berkeley DB releases, the DB_AUTO_COMMIT flag was used in the C and C++ Berkeley DB APIs to wrap operations within a transaction without explicitly creating a transaction and passing the TXN handle as part of the operation method call. In the 4.4 release, the DB_AUTO_COMMIT flag no longer needs to be explicitly specified.

In the 4.4 release, specifying the DB_AUTO_COMMIT flag to the DB_ENV->set_flags() method causes all database modifications in that environment to be transactional; specifying DB_AUTO_COMMIT to the DB->open() method causes all modifications to that database to be transactional; specifying DB_AUTO_COMMIT to the DB_ENV->dbremove() methods causes those specific operations to be transactional.

No related application changes are required for this release, as the DB_AUTO_COMMIT flag is ignored where it is no longer needed. However, application writers are encouraged to remove uses of the DB_AUTO_COMMIT flag in places where it is no longer needed.

Similar changes have been made to the Berkeley DB Tcl API. These changes are not optional, and Tcl applications will need to remove the -auto_commit flag from methods where it is no longer needed.

DB_DEGREE_2, DB_DIRTY_READ

The names of two isolation-level flags changed in the Berkeley DB 4.4 release. The DB_DEGREE_2 flag was renamed to DB_READ_COMMITTED, and the DB_DIRTY_READ flag was renamed to DB_READ_UNCOMMITTED, to match ANSI standard names for isolation levels. The historic flag names continue to work in this release, but may be removed from future releases.

DB_JOINENV

The semantics of joining existing Berkeley DB database environments has changed in the 4.4 release. Previously:

1. Applications joining existing environments, but not configuring some of the subsystems configured in the environment when it was created, would not be configured for those subsystems.

2. Applications joining existing environments, but configuring additional subsystems in addition to the subsystems configured in the environment when it was created, would cause additional subsystems to be configured in the database environment.

In the 4.4 release, the semantics have been simplified to make it easier to write robust applications. In the 4.4 release:

1. Applications joining existing environments, but not configuring some of the subsystems configured in the environment when it was created, will now automatically be configured for all of the subsystems configured in the environment.
2. Applications joining existing environments, but configuring additional subsystems in addition to the subsystems configured in the environment when it was created, will fail, as no additional subsystems can be configured for a database environment after it is created.

In other words, the choice of subsystems initialized for a Berkeley DB database environment is specified by the thread of control initially creating the environment. Any subsequent thread of control joining the environment will automatically be configured to use the same subsystems as were created in the environment (unless the thread of control requests a subsystem not available in the environment, which will fail). Applications joining an environment, able to adapt to whatever subsystems have been configured in the environment, should open the environment without specifying any subsystem flags. Applications joining an environment, requiring specific subsystems from their environments, should open the environment specifying those specific subsystem flags.

The DB_JOINENV flag has been changed to have no effect in the Berkeley DB 4.4 release. Applications should require no changes, although uses of the DB_JOINENV flag may be removed.

mutexes

The DB_ENV->set_tas_spins and DB_ENV->get_tas_spins methods have been renamed to DB_ENV->mutex_set_tas_spins() and DB_ENV->mutex_get_tas_spins() to match the new mutex support in the Berkeley DB 4.4 release. Applications calling the old methods should be updated to use the new method names.

For backward compatibility, the string "set_tas_spins" is still supported in DB_CONFIG files.

The --with-mutexalign="ALIGNMENT" compile-time configuration option has been removed from Berkeley DB configuration. Mutex alignment should now be configured at run-time, using the DB_ENV->mutex_set_align() method.

DB_MPOOLFILE->set_clear_len

The meaning of a 0 "clear length" argument to the DB_MPOOLFILE->set_clear_len() method changed in the Berkeley DB 4.4 release. In previous releases, specifying a length of 0 was equivalent to the default, and the entire created page was cleared. Unfortunately, this left no way to specify that no part of the page needed to be cleared. In the 4.4 release, specifying a "clear length" argument of 0 means that no part of the page need be cleared.

Applications specifying a 0 "clear length" argument to the `DB_MPOOLFILE->set_clear_len()` method should simply remove the call, as the default behavior is to clear the entire created page.

lock statistics

The names of two fields in the lock statistics changed in the Berkeley DB 4.4 release. The `st_nconflicts` field was renamed to be `st_lock_wait`, and the `st_nnowaits` field was renamed to be `st_lock_nowait`. The meaning of the fields is unchanged (although the documentation has been updated to make it clear what these fields really represent).

Upgrade Requirements

The log file format changed in the Berkeley DB 4.4 release. No database formats changed in the Berkeley DB 4.4 release.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Berkeley DB 4.4.16 Change Log

Database or Log File On-Disk Format Changes:

1. The on-disk log format has changed.

New Features:

1. Add support to compact an existing Btree database. [#6750]
2. Add support for named in-memory databases. [#9927]
3. Add support for database environment recovery serialization. This simplifies multiprocess application architectures. Add `DB_REGISTER` flag to `DB_ENV->open()`. [#11511]
4. Add utility for performing hot backups of a database environment. [#11536]
5. Add replication configuration API. [#12110]
6. Add replication support to return error instead of waiting for client sync to complete. [#12110]
7. Add replication support for delayed client synchronization. [#12110]
8. Add replication support for client-to-client synchronization. [#12110]
9. Add replication support for bulk transfer. [#12110]
10. Add new flags `DB_DSYNC_DB` and `DB_DSYNC_LOG` [12941]
11. Add `DbEnv.log_printf`, a new `DbEnv` method which logs `printf` style formatted strings into the Berkeley DB database environment log. [#13241]

Database Environment Changes:

1. Add a feature to support arbitrary alignment of mutexes in order to minimize cache line collisions. [#9580]
2. Change cache regions on 64-bit machines to allow regions larger than 4GB. [#10668]
3. Fix a bug where a loop could occur if the application or system failed during modification of the linked list of shared regions. [#11532]
4. Fix mutex alignment on Linux/PA-RISC, add test-and-set mutexes for MIPS and x86_64. [#11575]
5. Fix a bug where private database environments (DB_PRIVATE) on 64-bit machines would core dump because of 64-bit address truncation. [#11983]
6. Fix a bug where freed memory is accessed when DB_PRIVATE environments are closed. This can happen on systems where the operating system holds mutex resources that must be freed when the mutex is destroyed. [#12591]
7. Fix a bug where the DbEnv.stat_print method could self-deadlock and hang. The DbEnv.stat_print method no longer displays statistics for any of the database environments databases. [#12039]
8. Fix a bug where Berkeley DB could create fragmented filesystem-backed shared region files. [#12125]
9. Fix a bug where Berkeley DB stat calls could report a cache size of 0 after the statistics were cleared. [#12307]
10. Threads of control joining database environments are now configured for all of the subsystems (lock, log, cache, or transaction) for which the environment was originally configured, it is now an error to attempt configuration of additional subsystems after an environment is created. [#12422]
11. Fix a bug where negative percentages could be displayed in statistics output. [#12673]
12. Fix a bug that could cause a panic if the cache is filled with non-logging updated pages. [#12763]
13. Fix a bug that could cause an unreported deadlock if the application was using the DB_DIRTY_READ flag and the record was an off page duplicate record. [#12893]
14. Fix a bug where a handle lock could be incorrectly retained during a delete or rename operation. [#12906]

Concurrent Data Store Changes:

1. Lock upgrades and downgrades are now accounted for separately from lock requests and releases. [#11155]

2. Fix a bug where a second process joining a Concurrent Data Store environment, with the DB_CDB_ALLDB flag set, would fail. This would happen if the first thread were not entirely finished with initialization. [#12277]

General Access Method Changes:

1. Fix a bug where filesystem operations are improperly synchronized. [#10564]
2. Add support for database files larger than 2GB on Windows. [#11839]
3. Rename DB_DEGREE_2 (and all related flags) to DB_READ_COMMITTED. Rename DB_DIRTY (and all related flags) to DB_READ_UNCOMMITTED. [#11776]
4. Fix a bug where wrapping of sequences was incorrect when the cache size is smaller than the range of the maximum value minus the minimum value. [#11997]
5. Fix a bug that could result in a hot backup having a page missing from a database file if a file truncation was in progress during the backup but was then aborted. [#12017]
6. Fix a bug where a long filename could cause one too few bytes to be allocated when opening a file. [#12085]
7. Fix a bug in secondary cursor code if a write lock is not granted. [#12109]
8. Fix a bug in secondary cursors where the current record would change on error. [#12141]
9. Fix a bug in Db->truncate where the method was not checking to see if the handle was opened read-only. [#12179]
10. Fix a bug in sequences so that they are now platform independent, taking into account little-endian and big-endian architectures. They will be automatically upgraded in 4.4. [#12202]
11. Fix a bug with non-wrapping sequences when initial value was INT64_MIN. [#12390]
12. Add a retry for operating system operations that return EIO (IO Error) to better support NFS mounted filesystems. [#12426]
13. Fix sequence wrapping at INT64 limits. [#12520]
14. Fix a bug where errors during DB->associate could leave secondaries half associated. [#13173]
15. Fix a bug so that we no longer will update in CDS and DS if the file size limit will be exceeded. [#13222]

Btree Access Method Changes:

1. Remove maxkey configuration. [#8904]
2. Fix a memory leak in operations on large Btrees. [#12000]

Hash Access Method Changes:

1. Fix a bug where access to HASH or encrypted database pages might be blocked during a checkpoint. [#11031]
2. Fix a bug where recovery would fail when a database has a hash page on the free list and that hash page was freed without using transactions and later allocated and aborted within a transaction. [#11214]
3. Fix a bug in hash duplicates where if the caller left garbage in the partial length field, we were using it. Fix a bug where a replacement of a hash item that should have gone on an overflow page, did not. [#11966]
4. Fix a bug where free space was miscalculated when adding the first duplicate to an existing item and the existing item plus the new item does not fit on a page. [#12270]
5. Fix a bug where allocations of hash buckets are not recovered correctly. [#12846]

Queue Access Method Changes:

1. Improve performance of deletes from a QUEUE database that does not have a secondary index. [#11538]
2. Fix a bug where updates that do not use transactions, but do enable locking, failed to release locks. [#11669]
3. Fix a bug where a transaction might not be rolled forward if the site was performing hot backups and an application aborted a prepared but not committed transaction. [#12181]
4. Fix a bug with queue extents not being reclaimed. [#12249]
5. Fix a bug where a record being inserted before the head of the queue could appear missing if DB_CONSUME is not specified. [#12919]
6. Fix a bug that might cause recovery to move the head or tail of the queue to exclude a record that was deleted but whose transaction did not commit. [#13256]
7. Fix a bug that could cause recovery to move the head or tail pointer beyond a record that was aborted but was rolled backward by recovery. [#13318]

Recno Access Method Changes

None.

C++-specific API Changes:

1. Fix a bug so that a DbMemoryException will be raised during a DB_BUFFER_SMALL error. [#13273]

Java-specific API Changes:

1. Add VersionMismatchException to map the DB_VERSION_MISMATCH error. [#11429]

2. Fix a bug in `Environment.getConfiguration()` method in non-crypto builds. [#11752]
3. Fix a bug that caused a `NullPointerException` when using the `MultipleDataEntry` default constructor. [#11753]
4. Fix handling of replication errors. [#11822]
5. Remove `EnvironmentConfig.setReadOnly()` method. [#11882]
6. Fix a bug where prefix strings in the error handler may be corrupted. [#11967]
7. Fix a bug so that nested exceptions will appear in stack traces. [#11992]
8. Fix a bug on `LogSequenceNumber` objects in the Java API. [#12223]
9. Fix a bug when no files are returned from a call to `DB_ENV->log_archive`. [#12383]
10. Fix a bug when multiple verbose flags are set. [#12383]
11. Fix a bug so that an `OutOfMemoryError` is thrown when allocation fails in the JNI layer. [#13434]

Java collections and bind API Changes:

1. Binding performance has been improved by using `System.arraycopy` in the `FastOutputStream` and `FastInputStream` utility classes. [#12002]
2. The `objectToEntry` method is now implemented in all `TupleBinding` subclasses (`IntegerBinding`, etc) so that tuple bindings are fully nestable. An example of this usage is a custom binding that dynamically discovers the data types of each of the properties of a Java bean class. For each property, it calls `TupleBinding.getPrimitiveBinding` using the property's type (class). When the custom binding's `objectToEntry` method is called, it in turn calls the `objectToEntry` method of the nested bindings for each property. [#12124]
3. The `getCause` method for `IOExceptionWrapper` and `RuntimeExceptionWrapper` is now defined so that nested exceptions appear in stack traces for exceptions thrown by the collections API. [#11992]
4. `TupleBinding.getPrimitiveBinding` can now be passed a primitive type class as well as a primitive wrapper class. The return value for `Integer.TYPE` and `Integer.class`, for example, will be the same binding. [#12035]
5. Improvements have been made to prevent the buffer used in serial and tuple bindings from growing inefficiently, and to provide more alternatives for the application to specify the desired size. For details see `com.sleepycat.bind.serial.SerialBase` and `com.sleepycat.bind.tuple.TupleBase`. [#12398]
6. Add `StoredContainer.getCursorConfig`, deprecate `isDirtyRead`. Deprecate `StoredCollections.dirtyReadMap` (`dirtyReadSet`, etc) which is replaced by `configuredMap` (`configuredSet`, etc). Deprecate `StoredContainer.isDirtyReadAllowed` with no replacement (please use `DatabaseConfig.getDirtyRead`). Also note that

StoredCollections.configuredMap (configuredSet, etc) can be used to configure read committed and write lock containers, as well as read uncommitted containers, since all CursorConfig properties are supported. [#11776]

7. Add the protected method SerialBinding.getClassLoader so that subclasses may return a specific or dynamically determined class loader. Useful for applications which use multiple class loaders, including applications that serialize Groovy-defined classes. [#12764] [#12749]

Tcl-specific API Changes:

1. Fix a bug that could cause a memory leak in the replication test code. [#13436]

RPC-specific Client/Server Changes:

1. Fix double-free in RPC server when handling an out-of-memory error. [#11852]

Replication Changes:

1. Fix race condition (introduced in 4.3) in rep_start function. [#11030]
2. Changed internal initialization to no longer store records. [#11090]
3. Add support for replication bulk transfer. [#11099]
4. Berkeley DB now calls check_doreq function for MASTER_REQ messages. [#11207]
5. Fix a bug where transactions could be counted incorrectly during txn_recover. [#11257]
6. Add DB_REP_IGNORE flag so that old messages (especially PERM messages) can be ignored by applications. [#11585]
7. Fix a bug where op_timestamp was not initialized. [#11795]
8. Fix a bug in db_refresh where a client would write a log record on closing a file. [#11892]
9. Fix backward arguments in C++ rep_select API. [#11906]
10. Fix a bug where a race condition could happen between downgrading a master and a database update operation. [#11955]
11. Fix a bug on VERIFY_REQ. We now honor wait_recs/rcvd. [#12097]
12. Fix a bug in rebroadcast of verify_req by initializing lp->wait_recs when finding a new master. [#12097]
13. Fix a bug by adding lockout checking to __env_rep_enter since rename/remove now call it. [#12192]
14. Fix a bug so that we now skip __db_chk_meta if we are a rep client. [#12316]
15. Fix a replication failure on Windows. [#12331]

16. Remove master discovery phase from rep_elect as a performance improvement to speed up elections. [#12551]
17. Fix a bug to avoid multiple data streams when issuing al ALL_REQ. [#12595]
18. Fix a bug to request the remaining gap again if the gap record is dropped after we receive the singleton. [#12974]
19. Fix a bug in internal initialization when master changes in the middle of initializing. [#13074]
20. Fix a bug in replication/archiving with internal init. [#13110]
21. Fix pp handling of db_truncate. [#13115]
22. Fix a bug where rep_timestamp could be updated when it should not be updated. [#13331]
23. Fix a bug with bulk transfer when toggling during updates. [#13339]
24. Change EINVAL error return to DB_REP_JOIN_FAILURE. [#12110]
25. Add C++ exception for DB_REP_HANDLE_DEAD. [#13361]
26. Fix a bug where starting an election concurrently with processing a NEWMASTER message could cause the send function to be called with an invalid eid. [#13403]

XA Resource Manager Changes:

None.

Locking Subsystem Changes:

None.

Logging Subsystem Changes:

1. Add set_log_filemode for applications that need to set an absolute file mode on log files. [#8747]
2. Fix a bug that caused Not Found to be returned if a log file exists but is not readable. [#11185]
3. Removed checksum of records with an in-memory log buffer. [#11280]
4. Fix a bug so that the DB_LOG_INMEMORY flag can no longer be set after calling DB_ENV->open. [#11436]
5. Fix a bug introduced after release 4.0 where two simultaneous checkpoints could cause ckp_lsn values to be out of order. [#12094]
6. Fix a bug when in debug mode and using the DEBUG_ROP which will now log read operations in __dbc_logging. [#12303]

7. Fix a bug where failing to write a log record on a file close would result in a core dump later. [#12460]
8. Fix a bug where automatic log file removal, or the return of log files using an absolute path, could fail silently if the applications current working directory could not be reached using the systems getcwd library call. [#12505]
9. Avoid locking the log region if we are not going to flush the log. This can improve performance for some write-intensive application workloads. [#13090]
10. Fix a bug with a possible segment fault when memp_stat_print is called on a temporary database. [#13315]
11. Fix a bug where log_stat_print could deadlock with threads during a checkpoint. [#13315]

Memory Pool Subsystem Changes:

1. Fix a bug where modified database pages might not be flushed if recovery were run and all pages from a database were found in the system cache and up to date, followed by a system crash. [#11654]

Transaction Subsystem Changes:

1. Add new DbTxn class methods allowing applications to set/get a descriptive name associated with a transaction. The descriptive name is also displayed by the db_stat utility. [#0382]
2. Fix a bug where aborting a transaction with a large number of nested transactions could take a long time. [#10972]
3. Add support to allow the TXN_WRITE_NOSYNC flag to be specified on the transaction handle. [#11151]
4. Fix a bug that could cause a page to be on the free list twice if it was originally put on the free list by a non-transactional update and then reallocated in a transaction that aborts. [#11159]
5. Remove the requirement for the DB_AUTO_COMMIT flag to make database operations transactional. Specifying the database environment as transactional or opening the database handle transactionally is sufficient. [#11302]
6. Fix a bug so that environments created from errant programs that called dbp->close while transactions were still active can now be recovered. [#11384]
7. Fix a bug that caused free pages at the end of a file to be truncated during recovery rather than placed on the free page list. [#11643]
8. Fix a bug that caused a page to have the wrong type if the truncate of a BREE or RECNO database needed to be rolled forward. [#11670]
9. Fix a bug when manually undoing a subdb create, dont try to free a root page that has not been allocated. [#11925]

10. Add a check on database open to see if log files were incorrectly removed by system administration mistakes. [#12178]
11. Fix a bug when calling DB->pget and then specifying the DB_READ_COMMITTED (DB_DEGREE_2) on a cursor. If followed by a DBC->c_pget, the primary database would incorrectly remain locked. [#12410]
12. Fix a bug where the abort of a transaction in which a sub database was opened with the DB_TXN_NOT_DURABLE flag could fail. [#12420]
13. Fix a bug that could cause an abort transaction that allocated new pages to a file that were not flushed to disk prior to the abort transaction to report out of disk space. [#12743]
14. Fix a bug that could prevent multiple creates and destroys of the same file to be recovered correctly. [#13026]
15. Fix a bug when recovery previously handled a section of the log that did not contain any transactions. [#13139]
16. Fix a bug that could result in the loss of durability in Transactional Environments on Mac OS X. [#13149]
17. Fix a bug that could cause the improper reuse of a transaction id when recovery restores prepared transactions. [#13256]

Utility Changes:

1. Add utility for performing hot backups of a database environment. [#11536]
2. Change the Verify utility to now identify any nodes that have incorrect record counts. [#11934]
3. Fix a bug in the 1.85 compatibility code supporting per-application Btree comparison and prefix compression functions. The functions would not work on big-endian 64-bit hardware. [#13316]

Configuration, Documentation, Portability and Build Changes:

1. Change the ex_tpcb sample application to no longer displays intermediate results. It displays results at the end of the run. [#11259]
2. Change the Visual Studio projects on Windows so that each is in an intermediate directory. [#11441]
3. Fix errors in test subdb011. [#11799]
4. Fix a bug that could cause applications using gcc on Power PC platforms to hang. [#12233]
5. Fix a bug where installation will fail if a true program cannot be found. [#12278]
6. Fix a bug that prevented C++ applications from configuring XA [#12300].

7. Fix a race condition in the Windows mutex implementation found on 8-way Itanium systems. [#12417]
8. Add pthread mutex support for IBM OS/390 platform (z/OS or MVS). [#12639]
9. Fix a bug where the Tcl API did not configure on OS X 10.4. [#12699]
10. Fix portability issues with queue or recno primary databases. [#12872]
11. Fix a bug where utility attempted to send replication message. [#13446]

Berkeley DB 4.4.20 Change Log

Changes since Berkeley DB 4.4.16:

1. Add support for Visual Studio 2005. [#13521]
2. Fix a bug with in-memory transaction logs when files wrapped around the buffer. [#13589]
3. Fix a bug where we needed to close replications open files during replication initialization. [#13623]
4. Fix a bug which could leave locks in the environment if database compaction was run in a transactional environment on a non-transactional database. This might have also have triggered deadlocks if the database was opened transactionally. [#13680]
5. Fix a bug where setting the DB_REGISTER flag could result in unnecessarily running recovery, or corruption of the registry file on Windows systems. [#13789]
6. Fix a bug in Database.compact that could cause JVM crashes or NullPointerException. [#13791]
7. Fix a bug that would cause a trap if an environment was opened specifying DB_REGISTER and the environment directory could not be found. [#13793]
8. Fix a buffer overflow bug when displaying process and thread IDs in the Berkeley DB statistics output. [#13796]
9. Fix a bug where if there is insufficient memory for a database key in a DBT configured to return a key value into user-specified memory, the cursor is moved forward to the next entry in the database, which can cause applications to skip key/data pairs. [#13815]
10. Fix a bug that could cause the loss of an update to a QUEUE database in a hot backup. [#13823]
11. Fix a bug where retrieval from a secondary index could result in a core dump. [#13843]
12. Fix a bug that could cause part of the free list to become unlinked if a btree compaction was rolled back due to a transaction abort. [#13891]
13. Fix a bug with in-memory logging that could cause a race condition to corrupt the logs. [#13919]

Chapter 7. Upgrading Berkeley DB 4.2 applications to Berkeley DB 4.3

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 4.2 release interfaces to the Berkeley DB 4.3 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

Java

The Berkeley DB Java API has changed significantly in the 4.3 release, in ways incompatible with previous releases. This has been done to provide a consistent Java-like API for Berkeley DB as well as to make the Berkeley DB Java API match the API in Berkeley DB Java Edition, to ease application-porting between the two libraries.

Here is a summary of the major changes:

- Db -> Database
 - Dbc -> Cursor
 - Dbt -> DatabaseEntry
 - DbEnv -> Environment
 - DbTxn -> Transaction
 - Db.cursor -> Database.openCursor
 - Dbc.get(..., DbConstants.DB_CURRENT) -> Cursor.getCurrent(...)
1. The low-level wrapper around the C API has been moved into a package called `com.sleepycat.db.internal`.
 2. There is a new public API in the package `com.sleepycat.db`.
 3. All flags and error numbers have been eliminated from the public API. All configuration is done through method calls on configuration objects.
 4. All classes and methods are named to Java standards, matching Berkeley DB Java Edition. For example:
 5. The statistics classes have "getter" methods for all fields.
 6. In transactional applications, the Java API infers whether to auto-commit operations: if an update is performed on a transactional database without supplying a transaction, it is implicitly auto-committed.
 7. The `com.sleepycat.bdb.*` packages have been reorganized so that the binding classes can be used with the base API in the `com.sleepycat.db` package. The

bind and collection classes are now essentially the same in Berkeley DB and Berkeley DB Java Edition. The former `com.sleepycat.bdb.bind.*` packages are now the `com.sleepycat.bind.*` packages. The former `com.sleepycat.bdb`, `com.sleepycat.bdb.collections`, and `com.sleepycat.bdb.factory` packages are now combined in the new `com.sleepycat.collections` package.

8. A layer of the former collections API has been removed to simplify the API and to remove the redundant implementation of secondary indices. The former `DataStore`, `DataIndex`, and `ForeignKeyIndex` classes have been removed. Instead of wrapping a `Database` in a `DataStore` or `DataIndex`, the `Database` object is now passed directly to the constructor of a `StoredMap`, `StoredList`, etc.

DB_ENV->set_errcall, DB->set_errcall

The signature of the error callback passed to the `DB_ENV->set_errcall()` and `DB->set_errcall()` methods has changed in the 4.3 release. For example, if you previously had a function such as this:

```
void handle_db_error(const char *prefix, char *message);
```

it should be changed to this:

```
void handle_db_error(const DB_ENV *dbenv,  
    const char *prefix, const char *message);
```

This change adds the `DB_ENV` handle to provide database environment context for the callback function, and incidentally makes it clear the message parameter cannot be changed by the callback.

DBCursor->c_put

The 4.3 release disallows the `DB_CURRENT` flag to the `DBC->put()` method after the current item referenced by the cursor has been deleted. Applications using this sequence of operations should be changed to do the put without first deleting the item.

DB->stat

The 4.3 release adds transactional support to the `DB->stat()` method.

Application writers can simply add a `NULL txnid` argument to the `DB->stat()` method calls in their application to leave the application's behavior unchanged.

DB_ENV->set_verbose

The 4.3 release removes support for the `DB_ENV->set_verbose()` method flag `DB_VERB_CHKPOINT`. Application writers should simply remove any use of this flag from their applications.

The 4.3 release redirects output configured by the `DB_ENV->set_verbose()` method from the error output channels (see the `DB_ENV->set_errfile()` and `DB_ENV->set_errcall()` methods for

more information) to the new `DB_ENV->set_msgcall()` and `DB_ENV->set_msgfile()` message output channels. This change means the error output channels are now only used for errors, and not for debugging and performance tuning messages as well as errors. Application writers using `DB_ENV->set_verbose()` should confirm that output is handled appropriately.

Logging

In previous releases, the `DB_ENV->set_flags()` method flag `DB_TXN_NOT_DURABLE` specified that transactions for the entire database environment were not durable. However, it was not possible to set this flag in environments that were part of replication groups, and physical log files were still created. The 4.3 release adds support for true in-memory logging for both replication and non-replicated sites.

Existing applications setting the `DB_TXN_NOT_DURABLE` flag for database environments should be upgraded to set the `DB_LOG_INMEMORY` flag instead.

In previous releases, log buffer sizes were restricted to be less than or equal to the log file size; this restriction is no longer required.

DB_FILEOPEN

The 4.3 release removes the `DB_FILEOPEN` error return. Any application check for the `DB_FILEOPEN` error should be removed.

ENOMEM and DbMemoryException

In versions of Berkeley DB before 4.3, the error **ENOMEM** was used to indicate that the buffer in a DBT configured with `DB_DBT_USERMEM` was too small to hold a key or data item being retrieved. The 4.3 release adds a new error, `DB_BUFFER_SMALL`, that is returned in this case.

The reason for the change is that the use of **ENOMEM** was ambiguous: calls such as `DB->get()` or `DBC->get()` could return **ENOMEM** either if a DBT was too small or if some resource was exhausted.

The result is that starting with the 4.3 release, C applications should always treat **ENOMEM** as a fatal error. Code that checked for the **ENOMEM** return and allocated a new buffer should be changed to check for `DB_BUFFER_SMALL`.

In C++ applications configured for exceptions, a `DbMemoryException` will continue to be thrown in both cases, and applications should check the `errno` in the exception to determine which error occurred.

In Java applications, a `DbMemoryException` will be thrown when a `Dbt` is too small to hold a return value, and an `OutOfMemoryError` will be thrown in all cases of resource exhaustion.

Replication

The 4.3 release removes support for logs-only replication clients. Use of the `DB_REP_LOGONLY` flag to the `DB_ENV->rep_start()` should be replaced with the `DB_REP_CLIENT` flag.

The 4.3 release adds two new arguments to the `DB_ENV->rep_elect()` method, **nvotes** and **flags**. The **nvotes** argument sets the required number of replication group members that must participate in an election in order for a master to be declared. For backward compatibility, set the **nvotes** argument to 0. The **flags** argument is currently unused and should be set to 0. See `DB_ENV->rep_elect()` method or "Replication Elections" for more information.

In the 4.3 release it is no longer necessary to do a database environment hot backup to initialize a replication client. All that is needed now is for the client to join the replication group. Berkeley DB will perform an internal backup from the master to the client automatically and will run recovery on the client to bring it up to date with the master.

Run-time configuration

The signatures of the `db_env_set_func_ftruncate` and `db_env_set_func_seek` functions have been simplified to take a byte offset in one parameter rather than a page size and a page number.

Upgrade Requirements

The log file format changed in the Berkeley DB 4.3 release. No database formats changed in the Berkeley DB 4.3 release.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Berkeley DB 4.3.29 Change Log

Database or Log File On-Disk Format Changes:

1. The on-disk log format has changed.

New Features:

1. Add support for light weight, transactionally protected Sequence Number generation. [#5739]
2. Add support for Degree 2 isolation. [#8689]
3. Add election generation information to replication to support Paxos compliance. [#9068]
4. Add support for 64-bit and ANSI C implementations of the RPCGEN utility. [#9548]

Database Environment Changes:

1. Fix a bug where the permissions on system shared memory segments did not match the mode specified in the `DB_ENV->open()` method. [#8921]
2. Add a new return error from the `DB_ENV->open()` method call, `DB_VERSION_MISMATCH`, which is returned in the case of an application compiled under one version of Berkeley DB attempting to open an environment created under a different version. [#9077]

3. Add support for importing databases from a transactional database environment into a different environment. [#9324]
4. Fix a bug where a core dump could occur if a zero-length database environment name was specified. [#9233]
5. Increase the number of environment regions to 100. [#9297]
6. Remove the DB_ENV->set_verbose() method flag DB_VERB_CHKPOINT. [#9405]
7. Fix bugs where database environment getters could return incorrect information after the database environment was opened, if a different thread of control changed the database environment values. Fix bugs where database environment getter/setter functions could race with other threads of control. [#9724]
8. Change the DbEnv.set_lk_detect method to match the DbEnv.open semantics. That is, DbEnv.set_lk_detect may be called after the database environment is opened, allowing applications to configure automatic deadlock detection if it has not yet been configured. [#9724]
9. Fix cursor locks for environments opened without DB_THREAD so that they use the same locker ID. This eliminates many common cases of application self-deadlock, particularly in CDS. [#9742]
10. Fix a bug in DB->get_env() in the C API where it could return an error when it should only return the DB_ENV handle. C++ and Java are unchanged. [#9828]
11. Fix a bug where we only need to initialize the cryptographic memory region when MPOOL, Log or Transactions have been configured. [#9872]
12. Change private database environments at process startup to only allocate the heap memory required at any particular time, rather than always allocating the maximum amount of heap memory configured for the environment. [#9889]
13. Add a method to create nonexistent intermediate directories when opening database files. [#9898]
14. Add support for in-memory logging within database environments. [#9927]
15. Change Berkeley DB so configuring a database environment for automatic log file removal affects all threads in the environment, not just the DbEnv handle in which the configuration call is made. [#9947]
16. Change the signature of the error callback passed to the DB_ENV->set_errcall and DB->set_errcall methods to add a DB_ENV handle, to provide database environment context for the callback function. [#10025]
17. Fix a race condition between DB->close and DB->{remove, rename} on filesystems that don't allow file operations on open files (such as Windows). [#10180]
18. Add a DB_DSYNC_LOG flag to the DbEnv::set_flags method, which configures O_DSYNC on POSIX systems and FILE_FLAG_WRITE_THROUGH on Win32 systems. This offers

significantly better performance for some applications on certain Solaris/filesystem combinations. [#10205]

19. Fix a bug where calling the DB or DBEnv database remove or rename methods could cause a transaction checkpoint or cache flush to fail. [#10286]
20. Change file operations not to flush a file if it hasn't been written. [#10537]
21. Remove 4GB restriction on region sizes on 64 bit machines. [#10668]
22. Simplify the signature of substitute system calls for ftruncate and seek. [#10668]
23. Change Berkeley DB so that opening an environment without specifying a home directory will cause the DB_CONFIG file in the current directory to be read, if it exists. [#11424]
24. Fix a bug that caused a core dump if DB handles without associated database environments were used for database verification. [#11649]
25. Fix Windows mutexes shared between processes run as different users. [#11985]
26. Fix Windows mutexes for some SMP machines. [#12417]

Concurrent Data Store Changes:

1. Fix cursor locks for environments opened without DB_THREAD so that they use the same locker ID. This eliminates many common cases of application self-deadlock, particularly in CDS. [#9742]

General Access Method Changes:

1. Fix a bug where Berkeley DB log cursors would close and reopen the underlying log file each time the log file was read. [#8934]
2. Improve performance of DB->open() for existing subdatabases maintained within the same database file. [#9156]
3. Add a new error, DB_BUFFER_SMALL, to differentiate from ENOMEM. The new error indicates that the supplied DBT is too small. ENOMEM is now always fatal. [#9314]
4. Fix a bug when an update through a secondary index is deadlocked it is possible for the deadlock to be ignored, resulting in a partial update to the data. [#9492]
5. Fix a bug where a record could get inserted into the wrong database when a page was deallocated from one subdatabase and reallocated to another subdatabase maintained within the same database file. [#9510]
6. Enhance file allocation so that if the operating system supports decreasing the size of a file and the last page of the file is freed, it will be returned to the operating system. [#9620]
7. Fix a bug where DB_RUNRECOVERY could be returned if there was no more disk space while aborting the allocation of a new page in a database. [#9643]

8. Fix a bug where the cryptographic code could memcpy too many bytes. [#9648]
9. Fix a bug with DB->join() cursors that resulted in a memory leak and incomplete results. [#9763]
10. Disallow cursor delete, followed a cursor put of the current item across all access methods. [#9900]
11. Fix a bug where recovery of operations on unnamed databases that were never closed, could fail. [#10118]
12. Fix a bug where DB->truncate of a database with overflow records that spanned more than one page would loop. [#10151]
13. Improve performance in the database open/close path. [#10266]
14. Fix a bug that restricted the number of temporary files that could be created to 127. [#10415]
15. Fix a bug which could cause a Too many files error when trying to create temporary files. Limit the number of temporary file creation retries. [#10760] [#10773]
16. Fix a memory leak bug with Sequence Numbers. [#11589]
17. Fix a bug on Windows platforms that prevents database files from growing to over 2GB. [#11839]
18. Fix a platform independence bug with sequence numbers. Existing sequence numbers will be automatically upgraded upon next access. [#12202]
19. Fix a race between truncate and read/write operations on Windows platforms that could cause corrupt database files. [#12598]

Btree Access Method Changes:

1. Fix a bug where a record could get placed on the wrong page when two threads are simultaneously trying to split a four level (or greater) Btree. [#9542]
2. Fix a bug where calling DB->truncate() on a Btree which has duplicate keys that overflow the leaf page would not properly free the overflow pages and possibly loop. [#10666]

Hash Access Method Changes:

1. Fix a bug where a delete to a HASH database with off page duplicates could fail to have the proper lock when deleting an off page duplicate tree. [#9585]
2. Fix a bug where a dirty reader using a HASH database would leave a lock on the meta page. [#10105]
3. Fix a bug where a DB->del() on a HASH database supporting dirty reads could fail to upgrade a WWRITE lock to a WRITE lock when deleting an off page duplicate. [#10649]

Queue Access Method Changes:

1. Fix a bug where DB_CONSUME_WAIT may loop rather than wait for a new record to enter the queue, if the queue gets into a state where there are only deleted records between the head and the end of queue. [#9215]
2. Fix a bug where a Queue extent file could be closed when it was empty, even if a thread was still accessing a page from that file. [#9291]
3. Fix a bug where DBC->c_put(key, data, DB_CURRENT) where inserting a new record after the current record had been deleted was returning DB_KEYEMPTY. [#9314]
4. Fix a bug where a Queue extent file could be reported as not found if a race condition was encountered between removing the file and writing out a stale buffer. [#9462]
5. Fix a bug where the Queue access method might fail to release a record lock when running without transactions. [#9487]
6. Add DB_INORDER flag for Queue databases to guarantee FIFO (First In, First Out) ordering when using DB_CONSUME or DB_CONSUME_WAIT. [#9689]
7. Fix a bug where remove and rename calls could fail with a "Permission denied" error. [#9775]
8. Fix a bug where aborting a transaction that opened and renamed a queue database using extents could leave some of the extent files with the wrong name on Windows. [#9781]
9. Fix a bug where a db_dump of a queue database could return an error at the end of the queue if the head or tail of the queue is the first record on a page. [#10215]
10. Fix a race condition which would leave a Queue extent file open until the database handle was closed, preventing it from being removed. [#10591]
11. Fix a bug where a deadlock of a put on a database handle with dirty readers could generate a lock downgrade error. [#10678]
12. Fix a bug which caused DB_SET_RANGE and DB_GET_BOTH_RANGE to not return the next record when an exact match was not found. [#10860]

Recno Access Method Changes

1. Fix a bug where the key/data counts returned by the Db->stat method for Recno databases did not match the documentation. [#8639]
2. Fix a bug where DBC->c_put(key, data, DB_CURRENT) where inserting a new record after the current record had been deleted was returning DB_KEYEMPTY. [#9314].

C++-specific API Changes:

1. Change DbException to extend std::exception, making it possible for applications to catch all exceptions in one place. [#10022]

2. Fix a bug where errors during transaction destructors (commit, abort) could cause an invalid memory access. [#10302]
3. Fix a bug that could lead to a read through an uninitialized pointer when a DbLockNotGrantedException is thrown. [#10470]
4. Fix a bug in the C++ DbEnv::rep_elect method API where the arguments were swapped, leading to an "Invalid Argument" return when that method is called. [#11906]

Java-specific API Changes:

1. Fix a bug where the Java API did not respect non-zero return values from secondaryKeyCreate, including DB_DONOTINDEX. [#9474]
2. Fix a bug where a self-deadlock occurred with a non-transactional class catalog database used in a transactional environment. The bug only occurred when the collections API was not used for starting transactions. [#9521]
3. Improve Javadoc for the Java API. [#9614]
4. Improve memory management and performance when large byte arrays are being passed to DB methods. [#9801]
5. Improve performance of accessing statistics information from the Java API. [#9835]
6. Allow Java application to run without DB_THREAD so they can be used as RPC clients. [#10097]
7. Fix a bug where an uninitialized pointer is dereferenced for logArchive(Db.DB_ARCH_REMOVE). [#10225]
8. Fix a bug in the Collections API where a deadlock exception could leave a cursor open. [#10516]
9. Fix the replication callback in the Java API so that the parameter names match the C API. [#10550]
10. Add get methods to the Java statistics classes. [#10807]
11. Fix bugs in the Java API handling of null home directories and environments opened without a memory pool. [#11424]
12. Fix the Java API in the non-crypto package. [#11752]
13. Fix a bug that would cause corruption of error prefix strings. [#11967]
14. Fix handling of LSNs in the Java API. [#12223]
15. Dont throw a NullPointerException if the list of files returned by log_archive is empty. [#12383]

Tcl-specific API Changes:

None.

RPC-specific Client/Server Changes:

1. Add support for 64-bit and ANSI C implementations of the RPCGEN utility. [#9548]
2. Fix a small memory leak in RPC clients. [#9595]
3. Fix a bug in the RPC server to avoid self-deadlock by always setting DB_TXN_NOWAIT. [#10181]
4. Fix a bug in the RPC server so that if it times out an environment, it first closes all the Berkeley DB handles in that environment. [#10623]

Replication Changes:

1. Add an Environment ID to distinguish between clients from different replication groups. [#7786]
2. Add number of votes required and flags parameters to DB_ENV->rep_elect() method. [#7812]
3. Fix a bug where a client's env_openfiles pass could start with the wrong LSN. This could result in very long initial sync-up times for clients joining a replication group. [#8635]
4. Add election generation information to replication to support Paxos compliance. [#9068]
5. Add rep019 to test running normal recovery on clients to make sure we synch to the correct LSNs. [#9151]
6. Remove support for logs-only replication clients. Use of the DB_REP_LOGSONLY flag to the DB_ENV->rep_start() method should be replaced with the DB_REP_CLIENT flag. [#9331]
7. Fix a bug where replication clients fail to lock all the necessary pages when applying updates when there are more than one database in the transaction. [#9569]
8. Fix a bug in replication elections where when elections are called by multiple threads the wrong master could get elected. [#9770]
9. Fix a bug where the master could get a DB_REP_OUTDATED error. Instead send an OUTDATED message to the client. [#9881]
10. Add support for automatic initialization of replication clients. [#9927]
11. Modify replication timestamp so that non-replication client applications can get a DB_REP_HANDLE_DEAD. [#9986]
12. Add a new DB_REP_STARTUPDONE return value for rep_process_message() and st_startup_done to rep_stat() to indicate when a client has finished syncing up to a master and is processing live messages. [#10310]
13. Add _pp to secondary handles, add RPRINT, fix a deadlock. [#10429]
14. Fix a bug where an old client (and no master) that dropped the ALIVE message would never update to the current generation. [#10469]

15. Fix a bug where a message could get sent to a new client before NEWSITE has been returned to the application. Broadcast instead. [#10508]
16. Fix a crash when verbose replication messages are configured and a NULL DB_LSN pointer is passed to rep_process_message. [#10508]
17. Add code to respect set_rep_limit in LOG_REQ processing. [#10716]
18. Fix a synchronization problem between replication recovery and database open. [#10731]
19. Change elections to adjust timeout if egen changes while we are waiting. [#10686]
20. Client perm messages now return ISPERM/NOTPERM instead of 0. [#10855] [#10905]
21. Fix a race condition during rep_start when a role change occurs. Fix memory leaks. [#11030]
22. Fix problems with duplicate records. A failure will no longer occur if the records are old records (LOG_MORE) and archived. [#11090]
23. Fix a bug where the replication temporary database would grow during automatic client initialization. [#11090]
24. Add throttling to PAGE_REQ. [#11130]
25. Remove optimization-causing problems with racing threads in rep_verify_match. [#11208]
26. Fix memory leaks. [#11239]
27. Fix an initialization bug when High Availability configurations are combined with private database environments, which can cause intermittent failures. [#11795]
28. Fix a bug in the C++ DbEnv::rep_elect method API where the arguments were swapped, leading to an "Invalid Argument" return when that method is called. [#11906]

XA Resource Manager Changes:

None.

Locking Subsystem Changes:

1. Fix a bug where a deadlock of an upgrade from a dirty read to a write lock during an aborted transaction, may not be detected. [#7143]
2. Add support for Degree 2 isolation. [#8689]
3. Change the system to return DB_LOCK_DEADLOCK if a transaction attempts to get new locks after it has been selected as the deadlock victim. [#9111]
4. Fix a bug where when configured to support dirty reads, a writer may not downgrade a write lock as soon as possible, potentially blocking dirty readers. [#9197]
5. Change the test-and-set mutex implementation to avoid interlocked instructions when we know the instruction is unlikely to succeed. [#9204]

6. Fix a bug where a thread supporting dirty readers can get blocked while trying to get a write lock. It will allocate a new lock rather than using an existing WAS_WRITE lock when it becomes unblocked, causing the application to hang. [#10093]
7. The deadlock detector will now note that a parent transaction should be considered in abort if one of its children is. [#10394]
8. Remove a deadlock where database closes could deadlock with page acquisition. [#10726]
9. Fix a bug where a dirty reader could read an overflow page that was about to be deleted. [#10979]
10. Fix a bug that failed to downgrade existing write locks during a btree page split when supporting dirty reads. [#10983]
11. Fix a bug that would fail to upgrade a write lock when moving a cursor off a previously deleted record. [#11042]

Logging Subsystem Changes:

1. Fix a bug where recovery could leave too many files open. [#9452]
2. Fix a bug where aborting a transaction with a file open in it could result in an unrecoverable log file. [#9636]
3. Fix a bug where recovery would not return a fatal error if the transaction log was corrupted. [#9841]
4. Fix a bug in recovery so that the final checkpoint no longer tries to flush the log. This will permit recovery to complete even if there is no disk space to grow the log file. [#10204]
5. Improve performance of log flushes by pre-allocating log files and using fdatsync() in preference to fsync(). [#10228]
6. Fix a bug where recovery of a page split after a non-transactional update to the next page would fail to update the back pointer. [#10421]
7. Fix a bug in log_archive() where __env_rep_enter() was called twice. [#10577]
8. Fix a bug with in-memory logs that could cause a memory leak in the log region. [#11505]

Memory Pool Subsystem Changes:

1. Fix a bug in the MPOOLFILE file_written flag value so that checkpoint doesn't repeatedly open, flush and sync files in the cache for which there are no active application handles. [#9529]

Transaction Subsystem Changes:

1. Fix a bug where the same transaction ID could get allocated twice if you wrapped the transaction ID space twice and then had a very old transaction. [#9036]

2. Fix a bug where a transaction abort that contained a page allocation could loop if the filesystem was full. [#9461]
3. Fix implementation of DB->get_transactional() to match documentation: there is no possibility of error return, only 0 or 1. [#9526]
4. Fix a bug where re-setting any of the DB_TXN_NOSYNC, DB_TXN_NOT_DURABLE and DB_TXN_WRITE_NOSYNC flags could fail to clear previous state, potentially leading to incorrect transactional behavior in the application. [#9947]
5. Add a feature to configure the maximum number of files that a checkpoint will hold open. [#10026]
6. An aborting transaction will no longer generate an undetected deadlock. [#10394]
7. Fix a bug that prevented a child transaction from accessing a database handle that was opened by its parent transaction. [#10783]
8. Fix a bug where a checkpoint or a delayed write in another process could raise an EINVAL error if the database had been opened with the DB_TXN_NOT_DURABLE flag. [#10824]
9. Fix private transactional environments on 64-bit systems. [#11983]

Utility Changes:

1. Add debugging and performance tuning information to db_stat. Add new Berkeley DB handle methods to output debugging and performance tuning information to a C library FILE handle (C and C++ APIs only). [#9204]
2. Fix a bug where db_stat could drop core if DB->open fails and no subdatabase was specified. [#9273]
3. Add command-line arguments to the db_printlog utility to restrict the range of log file records that are displayed. [#9307]
4. Fix a bug in the locking statistics where current locks included failed lock requests. [#9314]
5. Fix a bug where db_archive would remove all log files when --enable-diagnostic and DB_NOTDURABLE were both specified. [#9459]
6. Fix a bug where db_dump with the -r flag would output extra copies of the subdatabase information. [#9808]
7. Fix a bug in db_archive that would cause log file corruption if the application had configured the environment with DB_PRIVATE. [#9841]
8. Add support in db_load for resetting database LSNs and file IDs without having to reload the database. [#9916]
9. Change the DB->stat() method to take a transaction handle as an argument, allowing DB->stat() to be called from within a transaction. [#9920]

10. Fix a bug in db_printlog where only the first file would be displayed for in-memory logs. [#11505]
11. Fix a bug that prevented database salvage from working in Berkeley DB 4.3.21. [#11649]
12. Fix a bug in db_load which made it impossible to specify more than a single option on the command line. [#11676]

Configuration, Documentation, Portability and Build Changes:

1. Add pread and pwrite to the list of system calls applications can replace at run-time. [#8954]
2. Add support for UTF-8 encoding of filenames on Windows. [#9122]
3. Remove C++ dependency on snprintf. Compilers on HP/UX 10.20 are missing header file support for snprintf(). [#9284]
4. Change Berkeley DB to not use the open system call flag O_DIRECT, unless DB configured using --enable-o_direct. [#9298]
5. Fix several problems with mutex alignment on HP/UX 10.20. [#9404]
6. Fix a memory leak when HAVE_MUTEX_SYSTEM_RESOURCES is enabled. [#9546]
7. Fix a bug in the sec002.tcl test for binary data. [#9626]
8. Fix a bug where filesystem blocks were not being zeroed out in the On-Time embedded Windows OS. [#9640]
9. Fix build problems with the Java API in Visual Studio .NET 2003. [#9701]
10. Add support for the gcc compiler on the Opteron platform. [#9725]
11. Add support for the small_footprint build option for VxWorks. [#9820]
12. Add support for linking of DLLs with MinGW. [#9957]
13. Remove the make target which builds the RPM package from the Berkeley DB distribution. [#10233]
14. Add a C++/XML example for ex_repquote. [#10380]
15. Fix a bug to link with lrt only if detected by configure (Mac OS X issue). [#10418]
16. Fix a bug and link Java and Tcl shared libraries with lpthread if required, for mutexes. [#10418]
17. Add support for building Berkeley DB on the HP NonStop OSS (Tandem) platform. [10483]
18. Change Berkeley DB to ignore EAGAIN from system calls. This fixed problems on NFS mounted databases. [#10531]

19. Remove a line with `bt_compare` and `bt_prefix` from the `db_dump` recovery test suite, which can cause failures on OpenBSD. [#10567]
20. Fix a conflict with the `lock_init` for building Berkeley DB on Cygwin. [#10582]
21. Add Unicode support for the Berkeley DB Windows API. [#10598]
22. Add support for 64-bit builds on Windows. [#10664]
23. Libtool version is now 1.5.8. [#10950]
24. Remove `mt` compilation flag for HP-UX 11.0. [#11427]
25. Fix a bug to link with `lrt` on Solaris to support `fdatasync`. [#11437]

Chapter 8. Upgrading Berkeley DB 4.1 applications to Berkeley DB 4.2

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 4.1 release interfaces to the Berkeley DB 4.2 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

Java

There are a number of major changes to the Java support in Berkeley DB in this release. Despite that we have tried to make this a bridge release, a release where we don't require you to change anything. We've done this using the standard approach to deprecation in Java. If you do not compile with deprecation warnings on, your existing sources should work with this new release with only minor changes despite the large number of changes. Expect that in a future release we will remove all the deprecated API and only support the new API names.

This is a list of areas where we have broken compatibility with the 4.1 release. In most cases it is a name change in an interface class.

- **DbAppDispatch.app_dispatch(DbEnv,Dbt,DbLsn,int)**
is now: **DbAppDispatch.appDispatch(DbEnv,Dbt,DbLsn,int)**
- **DbAppendRecno.db_append_recno(Db,Dbt,int)**
is now: **DbAppendRecno.dbAppendRecno(Db,Dbt,int)**
- **DbBtreeCompare.bt_compare(Db,Dbt,Dbt)**
is now: **DbBtreeCompare.compare(Db,Dbt,Dbt)**
- **DbBtreeCompare.dup_compare(Db,Dbt,Dbt)**
is now: **DbBtreeCompare.compareDuplicates(Db,Dbt,Dbt)**
- **DbBtreePrefix.bt_prefix(Db,Dbt,Dbt)**
is now: **DbBtreePrefix.prefix(Db,Dbt,Dbt)**
- **DbSecondaryKeyCreate.secondary_key_create(Db,Dbt,Dbt,Dbt)**
is now: **DbSecondaryKeyCreate.secondaryKeyCreate(Db,Dbt,Dbt,Dbt)**

The 4.2 release of Berkeley DB requires at minimum a J2SE 1.3.1 certified Java virtual machine and associated classes to properly build and execute. To determine what version virtual machine you are running, enter:

```
java -version
```

at a command line and look for the version number. If you need to deploy to a version 1.1 or 1.0 Java environment, it may be possible to do so by not including the classes in the `com.sleepycat.bdb` package in the Java build process (however, that workaround has not been tested by us).

A few inconsistent methods have been cleaned up (for example, `Db.close` now returns void; previously, it returned an int which was always zero). The synchronized attributed has been toggled on some methods -- this is an attempt to prevent multithreaded applications from calling close or similar methods concurrently from multiple threads.

The Berkeley DB API has up until now been consistent across all language APIs. Although consistency has its benefits, it made our Java API look strange to Java programmers. Many methods have been renamed in this release of the Java API to conform with Java naming conventions. Sometimes this renaming was simply "camel casing", sometimes it required rewording. The mapping file for these name changes is in `dist/camel.pm`. The Perl script we use to convert code to the new names is called `dist/camelize.pl`, and may help with updating Java applications written for earlier versions of Berkeley DB.

Berkeley DB has a number of places where as a C library it uses function pointers to move into custom code for the purpose of notification of some event. In Java the best parallel is the registration of some class which implements an interface. In this version of Berkeley DB we have made an effort to make those interfaces more uniform and predictable. Specifically, `DbEnvFeedback` is now `DbEnvFeedbackHandler`, `DbErrcall` is `DbErrorHandler` and `DbFeedback` is `DbFeedbackHandler`. In every case we have kept the older interfaces and the older registration methods so as to allow for backward compatibility in this release. Expect them to be removed in future releases.

As you upgrade to this release of Berkeley DB you will notice that we have added an entirely new layer inside the package `com.sleepycat.bdb`. This was formerly the Greybird project by Mark Hayes. Sleepycat Software and Mark worked together to incorporate his work. We have done this in hopes of reducing the learning curve when using Berkeley DB in a Java project. When you upgrade you should consider switching to this layer as over time the historical classes and the new `bdb` package classes will be more and more integrated providing a simple yet powerful interface from Java into the Berkeley DB library.

Berkeley DB's Java API is now generated with [SWIG](#). The new Java API is significantly faster for many operations.

Some internal methods and constructors that were previously public have been hidden or removed.

Packages found under `com.sleepycat` are considered different APIs into the Berkeley DB system. These include the core db api (`com.sleepycat.db`), the collections style access layer (`com.sleepycat.bdb`) and the now relocated XA system (`com.sleepycat.xa`).

Queue access method

We have discovered a problem where applications that specify Berkeley DB's encryption or data checksum features on Queue databases with extent files, the database data will not be protected. This is obviously a security problem, and we encourage you to upgrade these applications to the 4.2 release as soon as possible.

The Queue databases must be dumped and reloaded in order to fix this problem. First build the Berkeley DB 4.2 release, then use your previous release to dump the database, and the 4.2 release to reload the database. For example:

```
db-4.1.25/db_dump -P password -k database | \  
db-4.2.xx/db_load -P password new_database
```

Note this is **only** necessary for Queue access method databases, where extent files were configured along with either encryption or checksums.

DB_CHKSUM_SHA1

The flag to enable checksumming of Berkeley DB databases pages was renamed from DB_CHKSUM_SHA1 to DB_CHKSUM, as Berkeley DB uses an internal function to generate hash values for unencrypted database pages, not the SHA1 Secure Hash Algorithm. Berkeley DB continues to use the SHA1 Secure Hash Algorithm to generate hashes for encrypted database pages. Applications using the DB_CHKSUM_SHA1 flag should change that use to DB_CHKSUM; no other change is required.

DB_CLIENT

The flag to create a client to connect to a RPC server was renamed from DB_CLIENT to DB_RPCCLIENT, in order to avoid confusion between RPC clients and replication clients. Applications using the DB_CLIENT flag should change that use to DB_RPCCLIENT; no other change is required.

DB->del

In previous releases, the C++ Db::del and Java Db.delete() methods threw exceptions encapsulating the DB_KEYEMPTY error in some cases when called on Queue and Recno databases. Unfortunately, this was undocumented behavior.

For consistency with the other Berkeley DB methods that handle DB_KEYEMPTY, this is no longer the case. Applications calling the Db::del and Java Db.delete() methods on Queue or Recno databases, and handling the DB_KEYEMPTY exception specially, should be modified to check for a return value of DB_KEYEMPTY instead.

DB->set_cache_priority

In previous releases, applications set the priority of a database's pages in the Berkeley DB buffer cache with the DB->set_cache_priority method. This method is no longer available. Applications wanting to set database page priorities in the buffer cache should use the mempset_priority() method instead. The new call takes the same arguments and behaves identically to the old call, except that a DB_MPOOLFILE buffer cache file handle is used instead of the DB database handle.

DB->verify

In previous releases, applications calling the DB->verify() method had to explicitly discard the DB handle by calling the DB->close() method. Further, using the DB handle in other ways after

calling the `DB->verify()` method was not prohibited by the documentation, although such use was likely to lead to problems.

For consistency with other Berkeley DB methods, `DB->verify()` method has been documented in the current release as a DB handle destructor. Applications using the DB handle in any way (including calling the `DB->close()` method) after calling `DB->verify()` should be updated to make no further use of any kind of the DB handle after `DB->verify()` returns.

DB_LOCK_NOTGRANTED

In previous releases, configuring lock or transaction timeout values or calling the `DB_ENV->txn_begin()` method with the `DB_TXN_NOWAIT` flag caused database operation methods to return `DB_LOCK_NOTGRANTED`, or throw a `DbLockNotGrantedException` exception. This required applications to unnecessarily handle multiple errors or exception types.

In the Berkeley DB 4.2 release, with one exception, database operations will no longer return `DB_LOCK_NOTGRANTED` or throw a `DbLockNotGrantedException` exception. Instead, database operations will return `DB_LOCK_DEADLOCK` or throw a `DbDeadlockException` exception. This change should require no application changes, as applications must already be dealing with the possible `DB_LOCK_DEADLOCK` error return or `DbDeadlockException` exception from database operations.

The one exception to this rule is the `DB->get()` method using the `DB_CONSUME_WAIT` flag to consume records from a Queue. If lock or transaction timeouts are set, this method and flag combination may return `DB_LOCK_NOTGRANTED` or throw a `DbLockNotGrantedException` exception.

Applications wanting to distinguish between true deadlocks and timeouts can configure database operation methods to return `DB_LOCK_NOTGRANTED` or throw a `DbLockNotGrantedException` exception using the `DB_TIME_NOTGRANTED` flag.

The `DB_ENV->lock_get()` and `DB_ENV->lock_vec()` methods will continue to return `DB_LOCK_NOTGRANTED`, or throw a `DbLockNotGrantedException` exception as they have previously done.

Replication

Replication initialization

In the Berkeley DB 4.2 release, replication environments must be specifically initialized by any process that will ever do anything other than open databases in read-only mode (that is, any process which might call any of the Berkeley DB replication interfaces or modify databases). This initialization is done when the replication database environment handle is opened, by specifying the `DB_INIT_REP` flag to the `DB_ENV->open()` method.

Database methods and replication clients

All of the DB object methods may now return `DB_REP_HANDLE_DEAD` when a replication client changes masters. When this happens the DB handle is no longer able to be used and the application must close the handle using the `DB->close()` method and open a new handle. This

new return value is returned when a client unrolls a transaction in order to synchronize with the new master. Otherwise, if the application was permitted to use the original handle, it's possible the handle might attempt to access nonexistent resources.

DB_ENV->rep_process_message()

The DB_ENV->rep_process_message() method has new return values and an log sequence number (LSN) associated with those return values. The new argument is **ret_lsnp**, which is the returned LSN when the DB_ENV->rep_process_message() method returns DB_REP_ISPERM or DB_REP_NOTPERM. See Transactional guarantees for more information.

Client replication environments

In previous Berkeley DB releases, replication clients always behaved as if DB_TXN_NOSYNC behavior was configured, that is, clients would not write or synchronously flush their log when receiving a transaction commit or prepare message. However, applications needing a high level of transactional guarantee may need a write and synchronous flush on the client. By default in the Berkeley DB 4.2 release, client database environments write and synchronously flush their logs when receiving a transaction commit or prepare message. Applications not needing such a high level of transactional guarantee should use the environment's DB_TXN_NOSYNC flag to configure their client database environments to not do the write or flush on transaction commit, as this will increase their performance. Regardless of the setting of the DB_TXN_NOSYNC flag, clients will always write and flush on transaction prepare.

Tcl API

The Tcl API included in the Berkeley DB 4.2 release requires Tcl release 8.4 or later.

Upgrade Requirements

The log file format changed in the Berkeley DB 4.2 release. No database formats changed in the Berkeley DB 4.2 release.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Berkeley DB 4.2.52 Change Log

Database or Log File On-Disk Format Changes:

1. Queue databases that use encryption or data checksum features with extent files will need to be dumped and reloaded prior to using with release 4.2. For more details, see [Queue access method \(page 63\)](#). [#8671]
2. The on-disk log format changed.

New Features:

1. Add support for a reduced memory footprint build of the Berkeley DB library. [#1967]

2. Add the DB_MPOOLFILE->set_flags interface which disallows the creation of backing filesystem files for in-memory databases. [#4224]
3. Add cache interfaces to limit the number of buffers written sequentially to allow applications to bound the time they will monopolize the disk. [#4935]
4. Support auto-deletion of log files. [#0040] [#6252]
5. The new Java DBX API for Berkeley DB allows Java programmers to use a familiar Java Collections style API, including Map, while interacting with the transactional Berkeley DB core engine. [#6260]
6. Support auto-commit with the DB->get method's consume operations. [#6954]
7. Add "get" methods to retrieve most settings. [#7061]
8. Add Javadoc documentation to the Berkeley DB release. [#7110]
9. Add support to Concurrent Data Store to allow duplication of write cursors. [#7167]
10. Add C++ utility classes for iterating over multiple key and data items returned from a cursor when using the DB_MULTIPLE or DB_MULTIPLE_KEY flags. [#7351]
11. Add CamelCased methods to the Java API. [#7396]
12. Add the DB_MPOOLFILE->set_maxsize interface to enforce a maximum database size. [#7582]
13. Add a toString() method for all Java *Stat classes (DbBtreeStat, DbHashStat, DbMpoolStat, etc.). This method creates a listing of values of all of the class member variables. [#7712]

Database Environment Changes:

1. Add cache interfaces to limit the number of buffers written sequentially to allow applications to bound the time they will monopolize the disk. [#4935]
2. Fix a bug which could cause database environment open to hang, in database environments supporting cryptography. [#6621]
3. Fix a bug where a database environment panic might result from an out-of-disk-space error while rolling back a page allocation. [#6694]
4. Fix a bug where a database page write failure, in a database environment configured for encryption or byte-swapping, could cause page corruption. [#6791]
5. Fix a bug where DB->truncate could drop core if there were active cursors in the database. [#6846]
6. Fix a bug where for databases sharing a physical file required a file descriptor per database. [#6981]
7. Fix a bug where the panic callback routine was only being called in the first thread of control to detect the error when returning DB_RUNRECOVERY. [#7019]

8. Fix a bug where a transaction which contained a remove of a subdatabase and an allocation to another subdatabase in the same file might not properly be aborted. [#7356]
9. Fix a bug to now disallow DB_TRUNCATE on opens in locking environments, since we cannot prevent race conditions. In the absence of locking and transactions, DB_TRUNCATE will truncate ANY file for which the user has appropriate permissions. [#7345]
10. Fix several bugs around concurrent creation of databases. [#7363]
11. Change methods in DbEnv that provide access to statistics information so that they now return instances of the proper classes. [#7395]
12. Replace the DB->set_cache_priority API with the DB_MPOOLFILE->set_priority API. [#7545]
13. Fix a bug where a failure during a creation of a subdatabase could then fail in the dbremove cleanup, causing a crash. [#7579]
14. Allow creating into a file that was renamed within the same transaction. [#7581]
15. Fix a bug where DB_ENV->txn_stat could drop core if there are more-than-expected active transactions. [#7638]
16. Change Berkeley DB to ignore user-specified byte orders when creating a database in an already existing physical file. [#7640]
17. Fix a bug where a database rename that is aborted would leak some memory. [#7789]
18. Fix a bug where files could not be renamed or removed if they were not writable. [#7819]
19. Fix a bug where an error during a database open may leak memory in the mpool region. [#7834]
20. Fix a bug where the DB_ENV->trickle_sync method could flush all of the dirty buffers in the cache rather than a subset. [#7863]
21. Fix a bug where an attempt to rename or remove an open file in the same transaction could succeed, even though this is not allowed and will not work on Windows. [#7917]
22. Fix a bug where if a recovery interval in the log contained only database opens then a recovery might report "Improper file close". [#7886]
23. Add a flag, DB_INIT_REP to DB_ENV->open to initialize Replication subsystem. [#8299]
24. Fix a bug where file remove and rename operations would not block each other if they were in different transactions. [#8340]
25. Change Berkeley DB to not propagate error returns from the application's rep_send function out of the Berkeley DB API. [#8496] [#8522]
26. Remove restriction that DB_TRUNCATE is not allowed on files with subdatabases. This restriction was introduced in 4.1.25. [#8852]

Concurrent Data Store Changes:

1. Fix a bug where opens with other threads/processes actively acquiring locks on database handles could deadlock. [#6286]
2. Add support to Concurrent Data Store to allow duplication of write cursors. [#7167]

General Access Method Changes:

1. Fix a bug where the truncate of a database with associated secondary databases did not truncate the secondaries. [#6585]
2. Fix a bug in which an out-of-disk condition during a transactional database create, remove, or rename could cause a crash. [#6695]
3. Fix a bug where system errors unknown to the C library could cause Berkeley DB utilities to drop core on Solaris. [#6728]
4. Fix a bug where Berkeley DB could overwrite incorrectly formatted files rather than returning an error to the application during open. [#6769]
5. Fix a bug DB handle reference counts were incorrect, leading to spurious warning about open DB handles. [#6818]
6. Fix a bug where cursor adjustments across multiple DB handles could fail. [#6820]
7. Fix a bug where a failure during open could result in a hang. [#6902]
8. Fix a bug where repeated failures during certain stages of opens could cause error messages to appear during recovery. [#7008]
9. Fix a bug in secondary indices with multiple threads calling DBC->put that resulted in DB_NOTFOUND being returned. [#7124]
10. Fix a bug where database verification might reference memory which was previously freed after reporting an error. [#7137]
11. Rename the DB_CHKSUM_SHA1 to DB_CHKSUM as Berkeley DB only uses SHA1 for encrypted pages, not for clear text pages. [#7095]
12. Fix a bug where DB->rename could fail silently if the underlying system rename call failed. [#7322]
13. Fix a bug where Berkeley DB failed to open a file with FCNTL locking and 0-length files. [#7345]
14. Prohibit the use of the DB_RMW flag on get operations for DB handles opened in transactional mode. [#7407]
15. Standardize when Berkeley DB will return DB_LOCK_NOTGRANTED, or throw DbLockNotGrantedException, versus returning DB_LOCK_DEADLOCK or throwing DbDeadlockException. Fix bugs in the C++ and Java APIs where DbException was thrown,

encapsulating DB_LOCK_NOTGRANTED, rather than throwing DbLockNotGrantedException. [#7549]

16. Fix a bug where Berkeley DB could hang on a race condition if a checkpoint was running at the same time another thread was closing a database for the last time. [#7604]
17. Fix several bugs that made multiple filesystem level operations inside a single transaction break. [#7728]
18. Fix a memory leak in the abort path of a sub-database create. [#7790]
19. Fix a race condition with file close that could cause NULL pointer deference under load. [#8235]
20. Fix a bug to correct the calculation of the amount of space needed to return off page duplicates using the DB_MULTIPLE interface. [#8437]
21. Fix a bug where the duplicate data item count could be incorrect if a cursor was used to first overwrite and then delete a duplicate which was part of a set of duplicates large enough to have been stored outside the standard access method pages. [#8445]
22. Fix a bug where The DB_MULTIPLE interface might fail to return the proper duplicates in some edge cases. [#8485]
23. Fix a bug where DB->get(...DB_MULTIPLE) would not return a reasonable estimate of the buffer size required to return a set of duplicates. [#8513]
24. Fix a bug where the DbCursor.count method could return the wrong count in the case of small (on-page) duplicate sets, where a still-open cursor has been used to delete one of the duplicate data items. [#8851]
25. Fix a bug where a non-transactional cursor using DB_MULTIPLE_KEY could briefly be left pointing at an unlocked page. This could lead to a race condition with another thread deleting records resulting in the wrong record being deleted. [#8926]
26. Fix a bug where a key/data item could be lost if a cursor is used to do a delete, and then immediately used to do an insert which causes a set of duplicates to be shifted to an off-page Btree. [#9085]

Btree Access Method Changes:

1. Fix a bug where a deleted item could be left on a database page causing database verification to fail. [#6059]
2. Fix a bug where a page may be left pinned in the cache if a deadlock occurs during a DB->put operation. [#6875]
3. Fix a bug where a deleted record may not be removed from a Btree page if the page is split while another cursor is trying to delete a record on the page. [#6059]
4. Fix a bug where records marked for deletion were incorrectly counted when retrieving in a Btree by record number. [#7133]

5. Fix a bug where a page and lock were left pinned if an application requested a record number past the end of the file when retrieving in a Btree by record number. [#7133]
6. Fix a bug where deleted keys were included in the key count for the DB->stat call. [#7133]
7. Fix a bug where specifying MULTIPLE_KEY and NEXT_DUP to the bulk get interfaces might return the wrong data if all the duplicates could not fit in a single buffer. [#7192]
8. Remove assertions that triggered failures that were correct executions. [#8032]
9. Fix a bug where duplicate data items were moved onto overflow pages before it was necessary. [#8082]
10. Fix a bug where the DB->verify method might incorrectly complain about a tree's overflow page reference count. [#8061]
11. Fix a bug that could cause DB_MULTIPLE on a Btree database to return an incorrect data field at the end of buffer. [#8442]
12. Fix a bug where DBC->c_count was returning an incorrect count if the cursor was positioned on an item that had just been deleted. [#8851]
13. Remove the test for bt_maxkey in the Btree put code. If it is set to 1 it can cause an infinite loop. [#8904]

Hash Access Method Changes:

1. Fix a bug where Hash databases could be corrupted on filesystems that do not zero-fill implicitly created blocks. [#6588]
2. Fix a bug where creating a Hash database with an initial size larger than 4GB would fail. [#6805]
3. Fix a bug where a page in an unused hash bucket might not be empty if there was a disk error while writing the log record for the bucket split. [#7035]
4. Fix a bug where two threads opening a hash database at the same time might deadlock. [#7159]
5. Fix a bug where a hash cursor was not updated properly when doing a put with DB_NODUPDATA specified. [#7361]
6. Fix a bug that could cause DB_MULTIPLE_KEY on Hash databases to return improper results when moving from a key with duplicates to a key without duplicates. [#8442]

Queue Access Method Changes:

1. Fix a bug where opening an in-memory Queue database with extent size specified will dump core. [#6795]
2. Support auto-commit with the DB->get method's consume operations. [#6954]

3. Fix a bug where calling the sync method on a queue database with extents may hang if there are active consumers. [#7022]
4. Fix a bug where a get(...MULTIPLE...) might lead to an infinite loop or return the wrong record number(s) if there was a deleted record at the beginning of a page or the buffer was filled exactly at the end of a page. [#7064]
5. Fix a bug where a database environment checkpoint might hang if a thread was blocked waiting for a record while doing a DB_CONSUME_WAIT on a Queue database. [#7086]
6. Fix a bug where queue extent files would not be removed if a queue with extents was removed and its record numbers wrapped around the maximum record number. [#7191]
7. Fix a bug where a DB->remove of an extent based Queue with a small number of pages per extent would generate a segmentation fault. [#7249]
8. Fix a bug where verify and salvage on queues with extent files did not consider the extent files. [#7294]
9. Fix a bug when transaction timeouts are set in the environment they would get applied to some non-transactional operations and could cause a failure during the abort of a queue operation. [#7641]
10. Fix a bug when the record numbers in a queue database wrap around at 232, a cursor positioned on a record near the head of the queue that is then deleted, may return DB_NOTFOUND when get is specified with DB_NEXT rather than the next non-deleted record. [#7979]
11. Fix a bug where a record lock will not be removed when the first record in the queue is deleted without a transaction (not using DB_CONSUME). [#8434]
12. Fix a bug where byte swapping was not handled properly in queue extent files. [#8358]
13. Fix a bug where Queue extent file pages were not properly typed, causing the extent files not to use encryption or checksums, even if those options had been specified. This fix requires a database upgrade for any affected Queue databases. [#8671]
14. Fix a bug where truncating a queue with extents may fail to remove the last extent file. [#8716]
15. Fix a bug where a rename or remove of a QUEUE database with extents might leave empty extent files behind. [#8729]
16. Fix a bug where on Windows operating systems a "Permission denied" error may be raised if a Queue extent is reopened while it is in the process of being unlinked. [#8710]

Recno Access Method Changes:

1. Fix a bug where the DB->truncate method may return the wrong record count if there are deleted records in the database. [#6788]
2. Fix a bug where internal nodes of Recno trees could get wrong record count if a log write failed and the log was later applied during recovery. [#6841]

3. Fix a bug where a cursor next operation could infinitely loop after deleting a record, when the deleted record was immediately followed by implicitly created records. [#8133]

C++-specific API Changes:

1. Document the DB->del method can return DB_KEYEMPTY for Queue or Recno databases. The C++ and Java APIs now return this value rather than throwing an exception. [#7030]
2. Add "get" methods to retrieve most settings. [#7061]
3. Fix a bug where applications calling DB->verify from the C++ or Java APIs could drop core. Change the DB->verify method API to act as a DB handle destructor. [#7418]
4. Add utility classes for iterating over multiple key and data items returned from a cursor when using the DB_MULTIPLE or DB_MULTIPLE_KEY flags. These classes, DbMultipleDataIterator, DbMultipleKeyDataIterator, and DbMultipleRecnoDataIterator, mirror the DB Java API and are provided as replacements for the C macros, DB_MULTIPLE_INIT, DB_MULTIPLE_NEXT, DB_MULTIPLE_KEY_NEXT, and DB_MULTIPLE. [#7351]
5. Fix a bug DbException was thrown, encapsulating DB_LOCK_NOTGRANTED, rather than throwing DbLockNotGrantedException. [#7549]
6. Add the DbEnv handle to exceptions thrown by the C++ and Java APIs, where possible. [#7303]
7. Fix a bug in the C++ DbEnv::set_rep_transport signature so that the envid parameter is signed. [#8303]
8. Make the fields of DB_LSN public in the DbLsn class. [#8422]

Java-specific API Changes:

- a. Db.put(), Dbc.get() and Dbc.put() preserve key size
 - b. Dbc.get() returns DB_KEYEMPTY rather than throwing an exception
 - c. The return type of Db.close() is now void. [#7002]
1. New Java API (com.sleepycat.dbx.*) for the transactional storage of data using the Java Collections design pattern. [#6569]
 2. Fix a bug in the Java Dbt.get_recno_key_data() method when used inside callbacks. [#6668]
 3. Fix Java DbMpoolStat class to match the DB_MPOOL_STAT struct. [#6821]
 4. Fix a bug where Dbc.put expected key data even if the key was unused. [#6932]
 5. Fix a bug in the Java API secondary_key_create callback where memory was freed incorrectly, causing JVM crashes. [#6970]

6. Re-implement the Java API to improve performance and maintenance. Fix several inconsistencies in the Java API:
7. Document the DB->del method can return DB_KEYEMPTY for Queue or Recno databases. The C++ and Java APIs now return this value rather than throwing an exception. [#7030]
8. Add "get" methods to retrieve most settings. [#7061]
9. Add Javadoc documentation to the Berkeley DB release. [#7110]
10. Fix a bug that caused potential memory corruption when using the Java API and specifying the DB_DBT_REALLOC flag. [#7215]
11. Add the DbEnv handle to exceptions thrown by the C++ and Java APIs, where possible. [#7303]
12. Map existing c-style API to a more Java camel case API with Java style naming. Retained deprecated older API for the 4.2 release for backwards support in all cases except callback interfaces. Also overloaded methods such as get/pget() into multiple different get() calls to clean up call structure. [#7378]
13. Add CamelCased methods to the Java API. [#7396]
14. Fix a bug where applications calling DB->verify from the C++ or Java APIs could drop core. Change the DB->verify method API to act as a DB handle destructor. [#7418]
15. Fix a bug DbException was thrown, encapsulating DB_LOCK_NOTGRANTED, rather than throwing DbLockNotGrantedException. [#7549]
16. Add a toString() method for all Java *Stat classes (DbBtreeStat, DbHashStat, DbMpoolStat, etc.). This method creates a listing of values of all of the class member variables. [#7712]
17. Remove Db.fd() method from Java API as it has no value to a Java programmer. [#7716]
18. Add an accessible timeout field in the DbLockRequest class, needed for the DB_LOCK_GET_TIMEOUT operation of DbEnv.lockVector. [#8043]
19. Fix replication method calls from Java API. [#8467]
20. Fix a bug where exception returns were inconsistent. [#8622]
21. Change the Java API so that it throws an IllegalArgumentException rather than a DbException with the platform-specific EINVAL. [#8978]

Tcl-specific API Changes:

1. Add "get" methods to retrieve most settings. [#7061]
2. Brought Tcl's \$env set_flags command up to date with available flags. [#7385]
3. Update Berkeley DB to compile cleanly against the Tcl/Tk 8.4 release. [#7612]

4. Made txn_checkpoint publicly available. [#8594]

RPC-specific Client/Server Changes:

1. Fix two bugs in the RPC server where incorrect handling of illegal environment home directories caused server crashes. [#7075]
2. Fix a bug where the DB_ENV->close method would fail in RPC clients if the DB_ENV->open method was never called. [#8200]

Replication Changes:

1. Write prepare records synchronously on replication clients so that prepare operations are always honored in the case of failure. [#6416]
2. Change replication elections so that the client with the biggest LSN wins, and priority is a secondary factor. [#6568]
3. Fix a bug where replicas could not remove log files because the checkpoint lsn was not being updated properly. [#6620]
4. Force prepare records out to disk regardless of the setting of the DB_TXN_NOSYNC flag. [#6614]
5. Add a new flag, DB_REP_NOBUFFER, which gets passed to the rep_send function specified in DBENV->rep_set_transport, to indicate that the message should not be buffered on the master, but should be immediately transmitted to the client(s). [#6680]
6. Fix a replication election bug where Berkeley DB could fail to elect a master even if a master already existed. [#6702]
7. Allow environment wide setting of DB_AUTO_COMMIT on replication clients. [#6732]
8. Fix a replication bug where a client coming up in the midst of an election might not participate in the election. [#6826]
9. Add log_flushes when sites become replication masters. If log_flush fails, panic the environment since the clients already have the commits. [#6873]
10. Fix a replication bug where a brand new client syncing up could generate an error on the master. [#6927]
11. Fix a bug where clients synchronize with the master when they come up with the same master after a client-side disconnect or failures. [#6986]
12. Fix several bugs in replication elections turned up by test rep005. [#6990]
13. Fix a bug where aborted hash group allocations were not properly applied on replicas. [#7039]
14. Fix race conditions between running client recovery and other threads calling replication and other Berkeley DB functions. [#7402] [#8035]

15. Use shared memory region for all replication flags. [#7573]
16. Fix a bug where log archive on clients could prematurely remove log files. [#7659]
17. Return an error if a non-replication dbenv handle attempts to write log records to a replication environment. [#7752]
18. Fix a race condition when clients applied log records, where we would store a log record locally and then never notice we have it, and need to re-request it from the master, causing the client to get far behind the master. [#7765]
19. Fix inconsistencies between the documentation and actual code regarding when replication methods can be called. [#7775]
20. Fix a bug where Berkeley DB would wait forever if a NEWMASTER message got dropped. [#7897]
21. Fix a bug where the master environment ID did not get set when you called DBENV->rep_start as a master. [#7899]
22. Fix a bug where operations on a queue database will not get replicated if the transactions that include the operations are committed out of order with the operations. [#7904]
23. Fix bugs in log_c_get where an invalid LSN could access invalid addresses. Fix bug in elections where a client upgrading to master didn't write a txn_recycle record. [#7964]
24. Fix a bug where REP_VERIFY_FAIL during client recovery wasn't being handled. [#8040]
25. Return an error if the application calls rep_process_message before calling rep_start when starting. [#8057]
26. Fix a bug to ensure that replication generation numbers always increase and are never reset to 1. [#8136]
27. Modify log message retransmission protocol to efficiently handle the case where a large number of contiguous messages were dropped at once. [#8182] [#8169] [#8188]
28. Fix a bug where using the wrong mutex in replication which under certain conditions could cause replication to hang. Also fix a bug where incorrectly setting the checkpoint LSN could cause recovery to take a very long time. [#8183]
29. Fix bug where a message could get sent to an invalid master. [#8184]
30. Fix a bug where a local variable in log_archive was not initialized. [#8230]
31. Fix a bug where elections could hang. [#8254]
32. Fix a bug to ensure that we can always remove/re-create the temporary replication database after a failure. [#8266]
33. Add a flag, DB_INIT_REP to DB_ENV->open to initialize Replication subsystem. [#8299]

34. Add new `ret_lsn` argument to `rep_process_message` so that LSNs can be returned to clients on permanent records. Add new `lsnp` arg to the send callback function so that the master can know the LSNs of records as well. [#8308]
35. Narrow the window where we block due to client recovery. [#8316]
36. Fix a bug in `log_c_incursor` where we would not detect that a record was already in the buffer. [#8330]
37. Fix a bug that would allow elections to be managed incorrectly. [#8360]
38. Fix a bug where replicas were not maintaining `meta->last_pgno` correctly. [#8378]
39. Fix a bug in truncating log after recovery to a timestamp or replication-based recovery. [#8387]
40. Fix a bug where a checkpoint record written as the first record in a log could cause recovery to fail. [#8391]
41. Fix a bug where a client would return `DB_NOTFOUND` instead of `DB_REP_OUTDATED` when it was unable to synchronize with the master because it ran out of log records. [#8399]
42. Fix a bug where log file changes were not handled properly in replication. [#8400] [#8420]
43. Fix a bug where checking for invalid log header data could fail incorrectly. [#8460]
44. Fix a bug where `DB_REP_PERMANENT` was not being set when log records were re-transmitted. [#8473]
45. Modify elections so that all participants elect in the same election generation. [#8590]
46. Fix bug where `rep_apply` was masking an error return. Also return `DB_RUNRECOVERY` if the replication client cannot commit or checkpoint. [#8636]
47. Fix a bug to update the `last_pgno` on the meta page on free as well as alloc. [#8637]
48. Fix a bug to roll back the LSN on a queue database metapage if we're going to truncate the log. Fix a bug in `MASTER_CHECK` so we don't apply log messages from an unknown master. Fix a bug to perform a sync on `rep_close`. [#8601]
49. Fix a bug so that we reset the LSN when putting pages on the free list. [#8685]
50. Fix a bug where replication was not properly calling `db_shalloc`. [#8811]
51. Fix a bug where replication flags were getting set in multiple steps which could cause an Assertion Failure in `log_compare`. [#8889]
52. Fix a bug where open database handles could cause problems on clients. [#8936]
53. Fix a bug where in `dbreg` code where an `fnp` with an invalid `fileid` could be found on the `lp->fq` list. [#8963]
54. Fix a bug where a reader on a replication client could see partial updates when replicating databases with off page duplicates or overflow records. [#9041]

55. Fix a bug that could result in a self deadlock in dbreg under replication. [#9138]

56. Fix a memory leak in replication. [#9255]

XA Resource Manager Changes:

1. Fix a bug where a failed write during XA transaction prepare could result in a checksum error in the log. [#6760]
2. Fix a bug where we were not properly handling DB_AUTO_COMMIT in XA transactions and where we were not honoring the XA transaction during an XA-protected open. [#6851]
3. Add infrastructure support for multithreaded XA. [#6865]
4. Display XA status and ID as part of db_stat -t output. [#6413]

Locking Subsystem Changes:

- a. failure to remove dirty read locks prior to aborting a transaction,
 - b. calling upgrade on other than WWRITE locks,
 - c. failure to remove expired locks from the locker queue,
 - d. clearing the lock timeout before looking at it. [#7267]
1. Fix a bug where locks were not cleared in an off-page duplicate cursor. [#6950]
 2. Fix a bug where a deadlock may not be detected if dirty reads are enabled and the deadlock involves an aborting transaction. [#7143]
 3. Fix a bug where a transaction doing updates while using dirty read locking might fail while aborting the transaction with a deadlock. Several other locking issues were also fixed:
 4. Fix a bug when dirty reads are enabled a writer might be blocked on a lock that it had previously obtained. Dirty readers would also wait behind regular readers when they could have safely read a page. [#7502]
 5. Fix a bug where a DB->put using CDB gets a lock timeout then the error "Closing already closed cursor". [#7597]
 6. Modify the maximum test-and-set mutex sleep for logical page locks at 10ms, everything else at 25ms. [#7675]
 7. Fix a bug where the DB_LOCK_TIMEOUT mode of env->lock_vec could hang. [#7682]
 8. Fix a bug where running with only transaction timeouts for deadlock detection might deadlock without being detected if more than one transaction times out while trying to avoid searching a Btree on repeated inserts. [#7787]
 9. Fix a bug that could cause detection to not run when there was a lock that should be timed out. [#8588]

10. Fix a bug with using dirty reads with subtransactions. If a writing subtransaction aborts and then is blocked, the deadlock may not be detected. [#9193]
11. Fix a bug where handle locks were not being correctly updated when releasing read locks during transaction prepare. [#9275]

Logging Subsystem Changes:

1. Fix a bug where if a write error occurred while committing a transaction with DB_WRITE_NOSYNC enabled the transaction may appear to be committed in the log while it was really aborted. [#7034]
2. Fix a bug where multiprocess applications could violate write-ahead logging requirements if one process wrote a log record but didn't flush it, the current log file then changed, and another process wrote a database page before the log record was written to disk. [#6999]
3. Fix a bug where fatal recovery could fail with a "Transaction already committed" error if recovery had been run and there are no active transactions in the part of the log following the last checkpoint. [#7234]
4. Fix a bug where recovery would fail to put freed pages onto the free list, when both committed and aborted subtransactions that allocated new pages were present. This only affected prepared transactions. [#7403]
5. Fix a bug where open errors during recovery get propagated unless they are reporting missing files, which might correctly have been removed. [#7578]
6. Fix a bug so that we now validate a log file before writing to it. [#7580]
7. Fix a bug where Berkeley DB could display the unnecessary error message "DB_LOGC->get: short read" during recovery. [#7700]
8. Fix a bug where recovery may fail if it tries to reallocate a page to a file that is out of space. [#7780]
9. Change Berkeley DB so that operations on databases opened in a non-transactional mode do not write records into the database logs. [#7843]
10. Fix a bug where Berkeley DB could timeout waiting for locks (on Queue databases) during recovery. [#7927]
11. Fix a bug in truncating log after recovery to a timestamp or replication-based recovery. [#8387]
12. Fix a bug where recovery can be slow if the log contains many opens of files which contain multiple databases. [#8423]
13. Fix a bug where a file id could be used before its open was logged. [#8496]
14. Fix a bug where recovery would partially undo a database create if the transaction which created it spanned log files and not all of the log files were present during recovery. [#9039]

Memory Pool Subsystem Changes:

1. Fix a bug where checksummed files could not be read on different endian systems. [#6429]
2. Fix a bug where read-only databases were not mapped into memory but were instead read through the Berkeley DB buffer cache. [#6671]
3. Fix a bug where Berkeley DB could loop infinitely if the cache was sized so small that all of its pages were simultaneously pinned by the application. [#6681]
4. Fix a bug where DbEnv.sync could fail to write a page if another thread unpinned the page at the same time and there were no other pages in that hash bucket. [#6793]
5. Fix a bug where threads of control may hang if multiple threads of control are opening and closing a database at the same time. [#6953]
6. Fix a bug where a database created without checksums but later opened with checksums would result in a checksum error. [#6959]
7. Fix a bug where a multiprocess application suite could see incorrect data if one process opened a non-checksummed database
8. Change to avoid database open and flush when handles are discarded, if the handle was never used to write anything. [#7232]
9. Fix a bug where applications dirtying the entire cache in a single database operation would see large performance degradation. [#7273]
10. Fix a bug where contention in the buffer pool could cause the buffer allocation algorithm to unnecessarily sleep waiting for buffers to be freed. [#7572]

Transaction Subsystem Changes:

1. Fix a bug where disk write errors in encrypted database environments, causing transaction abort, could corrupt the log. [#6768]
2. Fix a bug where catastrophic recovery may fail on a log which has a prepared transaction which aborted the allocation of a new page and was rolled forward previously by another recovery session. [#6790]
3. Fix a bug where a transaction that contains a database truncate followed by page allocations, may not properly undo the truncate if aborted. [#6862]
4. Fix a bug which causes Berkeley DB to checkpoint quiescent database environments. [#6933]
5. Fix a bug where if a transaction prepare fails while writing the prepare log record, and it contains a subtransaction which did an allocation later, recovery of the database may fail with a log sequence error. [#6874]
6. Do not abort prepared but not yet completed transactions when closing an environment. [#6993]

7. Fix a bug where operations on the source of a rename in the same transaction would fail. [#7537]
8. Fix a bug where a parent transaction which aborts when it tries to write its commit record could fail with a log sequence error, if the parent transaction has an aborted child transaction which allocated a new page from the operating system. [#7251]
9. Fix a bug where Berkeley DB could try to abort a partial transaction because it contained a partial subtransaction. [#7922]
10. Fix a bug where Berkeley DB could drop core when transactions were configured without locking support. [#9255]

Utility Changes:

1. Fix a bug where db_load could core dump or corrupt record numbers by walking off the end of a string. [#6985]
2. Fix a bug where db_load could run out of locks when loading large numbers of records. [#7173]
3. Fix a bug where db_dump could drop core when salvaging unaligned entries on a Btree page. [#7247]
4. Fix a bug where hash statistics did not include overflow items in the count of database data elements. [#7473]
5. Fix a bug where an corruption in an overflow page list could cause DB->verify to infinitely loop. [#7663]
6. Fix a bug where verify could display extraneous error messages when verifying a Btree with corrupt or missing pages. [#7750]
7. Fix a bug that could cause the db_stat utility to display values larger than 100 for various percentages. [#7779]
8. Fix a memory overflow bug in db_load. [#8124]
9. Fix a minor leak when verifying queue databases. [#8620]

Configuration, Documentation, Portability and Build Changes:

1. Add support for a reduced memory footprint build of the Berkeley DB library. [#1967]
2. Change DB_SYSTEM_MEM on Windows to fail immediately when opening an environment whose regions were deleted on last close. [#4882]
3. Update queue.h to current FreeBSD version. [#5494]
4. Support for and certification under Tornado 2.2/VxWorks 5.5. [#5522]
5. Add support for IBM OS/390 using the IBM C compiler. [#6486]

6. Specify -pthread as a compile flag for Tru64 systems, not just as a linker flag. [#6637]
7. Remove automatic aggregate initialization for non-ANSI compilers. [#6664]
8. Fix a link error ("GetLongPathNameA could not be located in the dynamic link library KERNEL32.dll") that prevented Berkeley DB from loading on Windows NT. [#6665]
9. Remove use of U suffix in crypto build to denote unsigned integers for non-ANSI compilers. [#6663]
10. Fix Java API documentation problems where API return values were int and should have been void, or vice versa. [#6675]
11. Add an include of <sys/fcntl.h> for old Solaris systems with the directio call. [#6707]
12. Fix Java API documentation problem where the Db.associate call was missing a DbTxn handle. [#6714]
13. Clean up source based on gcc's -Wmissing-prototypes option. [#6759]
14. Ignore pread/pwrite interfaces on NCR's System V R 4.3 system. [#6766]
15. Fix an interface compatibility with Sendmail and Postfix releases. [#6769]
16. Fix warnings when the Tcl API was built without TEST_CONFIG defined. [#6789]
17. Change Win32 mutexes to use the shared code for all mutexes to fix handle leak. [#6822] [#6853]
18. Fix the Windows/Tcl API export list for Berkeley DB XML. [#6931]
19. Add the --enable-mingw configuration option to build Berkeley DB for MinGW. [#6973]
20. Add a CPU pause to the mutex spinlock code to improve performance on newer Pentium CPUs. [#6975]
21. Upgrade read-only file descriptors to read-write during checkpoint, it's an error to call FlushFileBuffers on a read-only Windows file handle. [#7051]
22. Fix configure so that Java applications on HP/UX can access RPC environments. [#7066]
23. Update Berkeley DB to use libtool 1.5 to allow building of shared libraries on various platforms. This should not be visible except for changes to the Makefile and internal build procedures. [#7080]
24. Fix a bug where the configure script displayed incorrect default installation directory information. [#7081]
25. Fix a signed/unsigned warning with some Windows compilers. [#7100]
26. Fix macro redefinition conflicts between queue.h and Vc7\PlatformSDK\Include\WinNT.h when building with Visual Studio.NET 7.0. [#7103]

-
27. Add a loop to retry system calls that return EBUSY. Also limit retries on EINTR to 100 times. [#7118]
 28. Fix a bug in our use of GetDiskFreeSpace that caused access violations on some versions of Windows with DB_DIRECT_DB. [#7122]
 29. Fix a bug where regions in system memory on Windows were incorrectly reinitialized because the magic number was overwritten. [#7127]
 30. Change version provided to Tcl's package system to reflect Berkeley DB's major and minor number. [#7174]
 31. Support for the Berkeley DB Embedix port has been removed. [#7209]
 32. Merge all public C++ headers into db_cxx.h, which fixes name clashes between Berkeley DB headers and system headers (specifically mutex.h). [#7221]
 33. Fix a bug where the configured Makefile could try and build objects for which there were no existing rules. [#7227]
 34. Port the ex_repquote example to Windows. [#7328]
 35. Fix a race in the ARM/gcc mutex code which could cause almost anything bad you can imagine. [#7468]
 36. Fix a bug where shared region removal could hang. [#7613]
 37. Fix a bug so that when using Java in Debug mode on Windows, automatically pick the Debug DLL. [#7722]
 38. Fix configure --disable-shared so that it now creates a Makefile that installs static libraries that look the same as a regular shared build. This flag will create a libdb<major>.<minor>.a and make a libdb.a that is a symlink to it. [#7755]
 39. Add support for OS/390 2.10 and all versions of z/OS. [#7972]
 40. Support Java builds on Windows with spaces in the project path. [#8141]
 41. Fix a bug where Berkeley DB mutex locking code for OS X was not multiprocessor safe. [#8255]
 42. Add an error to DB_ENV->set_flags if the OS does not support Direct
 43. Enable verbose error logging from the test suite on Windows. [#8634]
 44. Fix a bug with DLL linking on Cygwin under Windows. [#8628]
 45. Add support for JDK on HP/UX. [#8813]
 46. Fix a bug where pathnames longer than 2KB could cause processes to core dump. [#8886]
 47. Fix a bug in VxWorks when yielding the CPU, so that we delay at least one tick. [#9061]

Chapter 9. Upgrading Berkeley DB 4.0 applications to Berkeley DB 4.1

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 4.0 release interfaces to the Berkeley DB 4.1 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

DB_EXCL

The DB_EXCL flag to the DB->open() method now works for subdatabases as well as physical files, and it is now possible to use the DB_EXCL flag to check for the previous existence of subdatabases.

DB->associate, DB->open, DB->remove, DB->rename

Historic releases of Berkeley DB transaction-protected the DB->open(), DB->remove(), and DB->rename() methods, but did it in an implicit way, that is, applications did not specify the TXN handles associated with the operations. This approach had a number of problems, the most significant of which was there was no way to group operations that included database creation, removal or rename. For example, applications wanting to maintain a list of the databases in an environment in a well-known database had no way to update the well-known database and create a database within a single transaction, and so there was no way to guarantee the list of databases was correct for the environment after system or application failure. Another example might be the creation of both a primary database and a database intended to serve as a secondary index, where again there was no way to group the creation of both databases in a single atomic operation.

In the 4.1 release of Berkeley DB, this is no longer the case. The DB->open() and DB->associate() methods now take a TXN handle returned by DB_ENV->txn_begin() as an optional argument. New DB_ENV->dbremove() and DB_ENV->dbrename() methods taking a TXN handle as an optional argument have been added.

To upgrade, applications must add a TXN parameter in the appropriate location for the DB->open() method calls, and the DB->associate() method calls (in both cases, the second argument for the C API, the first for the C++ or Java APIs).

Applications wanting to transaction-protect their DB->open() and DB->associate() method calls can add a NULL TXN argument and specify the DB_AUTO_COMMIT flag to the two calls, which wraps the operation in an internal Berkeley DB transaction. Applications wanting to transaction-protect the remove and rename operations must rewrite their calls to the DB->remove() and DB->rename() methods to be, instead, calls to the new DB_ENV->dbremove() and DB_ENV->dbrename() methods. Applications not wanting to transaction-protect any of the operations can add a NULL argument to their DB->open() and DB->associate() method calls and require no further changes.

For example, an application currently opening and closing a database as follows:

```
DB *dbp;
DB_ENV *dbenv;
int ret;

if ((ret = db_create(&dbp, dbenv, 0)) != 0)
    goto err_handler;

if ((ret = dbp->open(dbp, "file", NULL, DB_BTREE,
    DB_CREATE, 0664)) != 0) {
    (void)dbp->close(dbp);
    goto err_handler;
}
```

could transaction-protect the DB->open() call as follows:

```
DB *dbp;
DB_ENV *dbenv;
int ret;

if ((ret = db_create(&dbp, dbenv, 0)) != 0)
    goto err_handler;

if ((ret = dbp->open(dbp,
    NULL, "file", NULL, DB_BTREE, DB_CREATE |
    DB_AUTO_COMMIT, 0664)) != 0) {
    (void)dbp->close(dbp);
    goto err_handler;
}
```

An application currently removing a database as follows:

```
DB *dbp;
DB_ENV *dbenv;
int ret;

if ((ret = db_create(&dbp, dbenv, 0)) != 0)
    goto err_handler;

if ((ret = dbp->remove(dbp, "file", NULL, 0)) != 0)
    goto err_handler;
```

could transaction-protect the database removal as follows:

```
DB *dbp;
DB_ENV *dbenv;
int ret;

if ((ret =
    dbenv->dbremove(dbenv, NULL, "file", NULL, DB_AUTO_COMMIT)) != 0)
    goto err_handler;
```

An application currently renaming a database as follows:

```
DB *dbp;
DB_ENV *dbenv;
int ret;

if ((ret = db_create(&dbp, dbenv, 0)) != 0)
    goto err_handler;

if ((ret = dbp->rename(dbp, "file", NULL, "newname", 0)) != 0)
    goto err_handler;
```

could transaction-protect the database renaming as follows:

```
DB *dbp;
DB_ENV *dbenv;
int ret;

if ((ret = dbenv->dbrename(
    dbenv, NULL, "file", NULL, "newname", DB_AUTO_COMMIT)) != 0)
    goto err_handler;
```

These examples are the simplest possible translation, and will result in behavior matching that of previous releases. For further discussion on how to transaction-protect DB->open() method calls, see Opening the databases.

DB handles that will later be used for transaction-protected operations must be opened within a transaction. Specifying a transaction handle to operations using handles not opened within a transaction will return an error. Similarly, not specifying a transaction handle to operations using handles that were opened within a transaction will also return an error.

DB_ENV->log_register

The DB_ENV->log_register and DB_ENV->log_unregister interfaces were removed from the Berkeley DB 4.1 release. It is very unlikely application programs used these interfaces. If your application used these interfaces, please contact us for help in upgrading.

st_flushcommit

The DB_ENV->log_stat "st_flushcommits" statistic has been removed from Berkeley DB, as it is now the same as the "st_scount" statistic. Any application using the "st_flushcommits" statistic should remove it, or replace it with the "st_count" statistic.

DB_CHECKPOINT, DB_CURLSN

The DB_CHECKPOINT flag has been removed from the DB_LOGC->get() and DB_ENV->log_put() methods. It is very unlikely application programs used this flag. If your application used this flag, please contact us for help in upgrading.

The DB_CURLSN flag has been removed from the DB_ENV->log_put() method. It is very unlikely application programs used this flag. If your application used this flag, please contact us for help in upgrading.

DB_INCOMPLETE

The DB_INCOMPLETE error has been removed from the 4.1 release, and is no longer returned by the Berkeley DB library. Applications no longer need to check for this error return, as the underlying Berkeley DB interfaces that could historically fail to checkpoint or flush the cache and return this error can no longer fail for that reason. Applications should remove all uses of DB_INCOMPLETE.

Additionally, the DbEnv.checkpoint and Db.sync methods have been changed from returning int to returning void.

DB_ENV->memp_sync

Historical documentation for the DB_ENV->memp_sync() method stated:
In addition, if DB_ENV->memp_sync() returns success, the value of **lsn** will be overwritten with the largest log sequence number from any page that was written by DB_ENV->memp_sync() to satisfy this request.

This functionality was never correctly implemented, and has been removed in the Berkeley DB 4.1 release. It is very unlikely application programs used this information. If your application used this information, please contact us for help in upgrading.

DB->stat.hash_nelem

The **hash_nelem** field of the DB->stat() method for Hash databases has been removed from the 4.1 release, this information is no longer available to applications.

Java exceptions

The Java DbEnv constructor is now marked with "throws DbException". This means applications must construct DbEnv objects in a context where DbException throwables are handled (either in a try/catch block or in a method that propagates the exception up the stack). Note that previous versions of the Berkeley DB Java API could throw this exception from the constructor but it was not marked.

C++ exceptions

With default flags, the C++ DbEnv and Db classes can throw exceptions from their constructors. For example, this can happen if invalid parameters are passed in or the underlying C structures could not be created. If the objects are created in an environment that is not configured for exceptions (that is, the DB_CXX_NO_EXCEPTIONS flag is specified), errors from the constructor will be returned when the handle's open method is called.

In addition, the behavior of the DbEnv and Db destructors has changed to simplify exception handling in applications. The destructors will now close the handle if the handle's close method was not called prior to the object being destroyed. The return value of the call is discarded, and no exceptions will be thrown. Applications should call the close method in normal situations so any errors while closing can be handled by the application.

This change allows applications to be structured as follows:

```
try {
    DbEnv env(0);
    env.open(/* ... */);
    Db db(&env, 0);
    db.open(/* ... */);
    /* ... */
    db.close(0);
    env.close(0);
} catch (DbException &dbe) {
    // Handle the exception, the handles have already been closed.
}
```

Application-specific logging and recovery

The application-specific logging and recovery tools and interfaces have been reworked in the 4.1 release to make it simpler for applications to use Berkeley DB to support their own logging and recovery of non-Berkeley DB objects. Specifically, the `DB_ENV->set_recovery_init` and `DB_ENV->set_tx_recover` interfaces have been removed, replaced by `DB_ENV->set_app_dispatch()`. Applications using either of the removed interfaces should be updated to call `DB_ENV->set_app_dispatch()`. For more information see Introduction to application specific logging and recovery and the `DB_ENV->set_app_dispatch()` documentation.

Upgrade Requirements

The log file format changed in the Berkeley DB 4.1 release.

All of the access method database formats changed in the Berkeley DB 4.1 release (Btree/Recno: version 8 to version 9, Hash: version 7 to version 8, and Queue: version 3 to version 4). **The format changes are entirely backward-compatible, and no database upgrades are needed.** Note that databases created using the 4.1 release may not be usable with earlier Berkeley DB releases.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Berkeley DB 4.1.24 and 4.1.25 Change Log

Database or Log File On-Disk Format Changes:

1. All of the access method database formats changed in the Berkeley DB 4.1 release (Btree/Recno: version 8 to version 9, Hash: version 7 to version 8, and Queue: version 3 to version 4). *The format changes are entirely backward-compatible, and no database upgrades are needed.*

Major New Features:

1. Berkeley DB now includes support for database encryption using the AES encryption standard. [#1797]

2. Berkeley DB now includes support for database page checksums to allow detection of database corruption during I/O. [#1797]
3. The shared memory buffer pool code base was substantially reworked in the 4.1 release to improve concurrent throughput. [#4655]

General Environment Changes:

1. Allow applications to specify transaction handles to the DB->open method call, so database creation can be grouped with other Berkeley DB calls in a single transaction. [#4257]
2. Add the DB_ENV->remove and DB_ENV->rename method calls that support transactional protection of database removal and renaming. [#4257]
3. Add the DB_ENV->set_flags flags DB_DIRECT_DB and DB_DIRECT_LOG, which disable the system's buffer cache where possible. [#4526]
4. Unlock the pthread mutex if pthread_cond_wait() returns an error. [#4872]
5. Fix a memory leak caused by running recovery. [#4913]
6. Fix a bug in which closing an environment with open database handles could result in application crashes. [#4991]
7. Fix a bug where DB_CONFIG files were ignored if the database environment defaulted to the application's current working directory. [#5265]
8. Fix a bug where transaction abort or commit could fail to destroy the handle. [#5633]
9. Fix a set of bugs where the Berkeley DB API could return DB_RUNRECOVERY without panicking the database environment itself or calling the application's panic-callback function. [#5743]
10. Fix a bug in where DB=>rename and DB->remove method calls could leak a transaction and its locks. [#5824]
11. Fix a bug where recovery feedback could return values greater than 100. [#6193]
12. Fix a bug where a page allocated by a transaction, eventually aborted because of application or system failure, could appear twice in the free list, if catastrophic recovery was performed. [#6222]
13. Add a new flag, DB_AUTO_COMMIT, that wraps all database modification operations inside a transaction, to the DB_ENV->set_flags method. [#6395]
14. Fix a bug where recovery could fail when upgrading between releases. [#6372]
15. Fix a recovery bug where pages that were repeatedly freed and allocated could be lost. [#6479] [#6501]
16. Change DB_CONFIG reading to handle non-<newline> terminated last line. [#6490]

General Access Method Changes:

1. Allow applications to specify transaction handles to the DB->associate method call, so secondary index creation can be grouped with other Berkeley DB calls in a single transaction. [#4185]
2. Add a new flag, DB_AUTO_COMMIT, that wraps single database operations inside a transaction. This flag is supported by the DB->del, DB->open, DB->put, DB->truncate, DB_ENV->remove, and DB_ENV->rename methods. [#4257]
3. The DB_EXCL DB->open method flag has been enhanced to work on subdatabases. [#4257]
4. Fix a bug in which a DB->put(DB_APPEND) could result in leaked memory or a corruption in the returned record number. [#5002]
5. Fix a bug in the database salvage code that could leave pages pinned in the cache. [#5037]
6. Add a flag to the DB->verify method to output salvaged key/data pairs in printable characters. [#5037]
7. Fix a bug in which DB->verify() might continue and report extraneous database corruption after a fatal error. [#5131]
8. Fix a bug where calling the DB->stat method before the DB->open method could drop core. [#5190]
9. Fix a bug in which a DB->get, DBcursor->c_get, or DBcursor->c_pget on a secondary index, in the Concurrent Data Store product, could result in a deadlock. [#5192]
10. Fix a bug in which DB->verify() could correctly report errors but still return success. [#5297]
11. Add support for the DB->set_cache_priority interface, that allows applications to set the underlying cache priority for their database files. [#5375]
12. Fix a bug where calling DBcursor->c_pget with a database that is not a secondary index would drop core. [#5391]
13. Fix a bug where a bug in the DB->truncate method could cause recovery to fail. [#5679]
14. Fix a bug where DB_GET_RECNO would fail if specified to a secondary index. [#5811]
15. Fix a bug where building a secondary index for an existing primary database could fail in Concurrent Data Store environments. [#5811]
16. Fix a bug where the DB->rename method could fail, causing a problem during recovery. [#5893]
17. Fix a bug in which a DB->get or DB->pget call on a secondary index could fail when done with a handle shared among multiple threads. [#5899]

18. Fix a bug in which a DB->put operation on a database with off-page duplicates could leak a duplicate cursor, thereby preventing transactions being able to commit. [#5936]
19. Fix a bug where overflow page reference counts were not properly maintained when databases were truncated. [#6168]
20. Fix a bug where the bulk get APIs could allocate large amounts of heap memory. [#6439] [#6520]

Btree Access Method Changes:

1. Fix a bug that prevented loads of sorted data, with duplicates at the end of the tree, from creating compact trees. [#4926]
2. No longer return a copy of the key if the DB_GET_BOTH or DB_GET_BOTH_RANGE flags are specified. [#4470]
3. Fix a bug where the fast-search code could hold an unlocked reference to a page, which could lead to recovery failure. [#5518]
4. Fix a bug where some cursor operations on a database, for which the bt_minkey size had been specified, could fail to use the correct overflow key/data item size. [#6183]
5. Fix a bug where the recovery of an aborted transaction that did a reverse Btree split might leave a page in an inconsistent state. [#6393]

Hash Access Method Changes:

1. Fix bugs that could cause hash recovery to drop core. [#4978]
2. Use access method flags instead of interface flags to check for read-only access to a hash database with an application-specified hash function. [#5121]
3. Fix a bug where a hash database allocation of a new set of buckets may be improperly recovered by catastrophic recovery if the transaction is split across log files and the beginning segment of the transaction is not included in the set of logs to be recovered. [#5942]
4. Fix a bug where aborting particular hash allocations could lead to a database on which the verifier would loop infinitely. [#5966]
5. Fix a bug where a memory allocation failure could result in a system hang. [#5988]
6. Remove nelem from the Hash access method statistics (the value was incorrect once items had been added or removed from the database). [#6101]
7. Fix a bug where a page allocated by an aborted transaction might not be placed on the free list by recovery, if the file holding the page was created as part of recovery, and a later page was part of a hash bucket allocation. [#6184]
8. Fix a bug where allocated pages could be improperly recovered on systems that require explicit zero-ing of filesystem pages. [#6534]

Queue Access Method Changes:

1. No longer return a copy of the key if the DB_SET_RANGE flag is specified. [#4470]
2. Fix a bug where Dbcursor->c_get (with DB_MULTIPLE or DB_MULTIPLE_KEY specified) could fail on a Queue database if the record numbers had wrapped. [#6397]

Recno Access Method Changes:

1. No longer return a copy of the key if the DB_GET_BOTH or DB_GET_BOTH_RANGE flags are specified. [#4470]
2. Fix a bug where non-transactional locking applications could leak locks when modifying Recno databases. [#5766]
3. Fix a bug where Dbcursor->c_get with the DB_GET_RECNO flag would panic the environment if the cursor was uninitialized. [#5935]
4. Fix a bug where deleting pages from a three-level Recno tree could cause the database environment to panic. [#6232]

C++-specific API Changes:

1. C++ DbLock::put is replaced by DbEnv::lock_put to match the C and Java API change in Release 4.0. [#5170]
2. Declared destructors and methods within Db and DbEnv classes to be virtual, making subclassing safer. [#5264]
3. Fixed a bug where Dbt objects with no flags set would not be filled with data by some operations. [#5706]
4. Added DbDeadlockException, DbRunRecoveryException, and DbLockNotGrantedException classes to C++, and throw them accordingly. [#6134]
5. Added C++ methods to support remaining conversions between C++ classes and C structs where appropriate. In particular, DbTxn/DB_TXN conversions and DbMpoolFile/DB_MPOOLFILE were added. [#6278]
6. Fix a bug in DbEnv::~DbEnv() that could cause memory corruption if a DbEnv was deleted without being closed. [#6342]
7. Reordered C++ class declarations to avoid a GCC g++ warning about function inlining. [#6406]
8. Fix a bug in the DbEnv destructor that could cause memory corruption when an environment was destroyed without closing first. [#6342]
9. Change DbEnv and Db destructor behavior to close the handle if it was not already closed. [#6342]

Java-specific API Changes:

1. Added check for system property "sleepycat.Berkeley DB.libfile" that can be used to specify a complete pathname for the JNI shared library. This is needed as a workaround on Mac OS X, where libtool cannot currently create a library with a .jnilib extension which is what the current JDK expects by default. [#5664]
2. Fixed handling of JVM out of memory conditions, when some JNI methods return NULL. When the JVM runs out of memory, calls should consistently fail with OutOfMemoryErrors. [#5995]
3. Added Dbt.get_object and Dbt.set_object convenience routines to the Java API to make using serialization easier. [#6113]
4. Fixed a bug that prevented Java's Db.set_feedback from working, fixed document for Java's Db.set_feedback, some callback methods were misnamed. [#6137]
5. Fix a NullPointerException in Db.finalize() if the database had been closed. [#6504]
6. Marked DbEnv constructor with "throws DbException". [#6342]

Tcl-specific API Changes:

None.

RPC-specific Client/Server Changes:

1. Fix a bug where Db and DbEnv handles were not thread-safe. [#6102]

Replication Changes:

1. A large number of replication bugs were fixed in this release. The replication support is now believed to be production quality.
2. Add the DB_ENV->set_rep_limit interface, allowing applications to limit the data sent in response to a single DB_ENV->rep_process_message call. [#5999]
3. Add the DB_ENV->set_rep_stat interface, returning information from the replication subsystem [#5919]

XA Resource Manager Changes:

1. Added support for multithreaded XA. Environments can now have multiple XA transactions active. db_env_xa_attach() can be used to get a DB_TXN that corresponds to the XA transaction in the current thread. [#5049]
2. Added a com.sleepycat.Berkeley DB.xa package that implements J2EE support for XA. This includes new DbXAResource, DbXid classes that implement the XAResource and Xid interfaces. [#5049]
3. Fix a bug where aborting a prepared transaction after recovery may fail. [#6383]

4. Fix a bug where recovery might fail if a prepared transaction had previously extended the size of a file and then was aborted. [#6387]
5. Fix a bug where if the commit of a prepared transaction fails the transaction would be aborted. [#6389]

Locking Subsystem Changes:

1. Fix a bug where lock counts were incorrect if a lock request returned DB_LOCK_NOTGRANTED or an error occurred. [#4923]
2. Fix a bug where lock downgrades were counted as releases, so the lock release statistics could be wrong. [#5762]
3. Fix a bug where the lock and transaction timeout values could not be reset by threads of control joining Berkeley DB database environments. [#5996]
4. Fix a bug where applications using lock and/or transaction timeouts could hit a race condition that would lead to a segmentation fault. [#6061]

Logging Subsystem Changes:

1. DB_ENV->log_register and DB_ENV->log_unregister have been removed from the interface. [#0046]
2. Fix a bug where creating a database environment with a nonexistent logging directory could drop core. [#5833]
3. Add support allowing applications to change the log file size in existing database environments. [#4875]
4. Fix a bug where a write error on a log record spanning a buffer could cause transaction abort to fail and the database environment to panic. [#5830]

Memory Pool Subsystem Changes:

1. The DB_INCOMPLETE error has been removed, as cache flushing can no longer return without completing. [#4655]
2. Fix a bug where Berkeley DB might refuse to open a file if the open was attempted while another thread was writing a large buffer. [#4885]
3. Prefer clean buffers to dirty buffers when selecting a buffer for eviction. [#4934]
4. Fix a bug where transaction checkpoint might miss flushing a buffer to disk. [#5033]
5. Fix a bug where Berkeley DB applications could run out of file descriptors. [#5535]
6. Fix bugs where Berkeley DB could self-deadlock on systems requiring mutex resource reclamation after application failure. [#5722] [#6523]

Transaction Subsystem Changes:

1. Go back only one checkpoint, not two, when performing normal recovery. [#4284]
2. Fix a bug where an abort of a transaction could fail if there was no disk space for the log. [#5740]
3. Fix a bug where the checkpoint log-sequence-number could reference a nonexistent log record. [#5789]
4. Fix a bug where subtransactions which allocated pages from the filesystem and subsequently aborted could cause other pages allocated by sibling transactions to not be freed if the parent transaction then aborted. [#5903]
5. Fix a bug where transactions doing multiple updates to a queue database which spanned a checkpoint could be improperly handled by recovery. [#5898]

Utility Changes:

1. Fix a bug where the -p option could not be specified with the -R or -r options. [#5037]
2. The utilities were modified to correctly size their private caches in order to handle databases with large page sizes. [#5055]
3. Fix a bug in which utilities run with the -N option would fail to ignore the environment's panic flag. [#5082]
4. Fix a bug where invalid log records could cause db_printlog to drop core. [#5173]
5. Add a new option to the db_verify utility to support verification of files that include databases having non-standard sorting or hash functions. [#5237]

Configuration, Documentation, Portability and Build Changes:

1. Replace test-and-set mutexes on Windows with a new mutex implementation that signals an event to wake blocked threads. [#4413]
2. Support configuration of POSIX pthread mutexes on systems where the pthread mutexes do not support inter-process locks. [#4942]
3. Add mutex support for the ARM architecture using the gcc compiler. [#5018]
4. On Windows NT/2000/XP, switched to atomic seek-and-read/write operations to improve performance of concurrent reads [#0654].
5. Support cross-compilation using the GNU compiler tool chain. [#4558]
6. Fix a bug where libraries were always installed read-only. [#5096]
7. Fix a bug where temporary files on VxWorks could fail. [#5160]
8. Fix a bug where Berkeley DB did not install correctly if the system cp utility did not support the -f option. [#5111]

-
9. Correct the documentation for the Queue access method statistics field `qs_cur_recno` to be the "Next available record number". [#5190]
 10. Fix a bug where file rename could fail on Windows/9X. [#5223]
 11. Removed support for Microsoft Visual Studio 5.0 [#5231]
 12. Switched to using `HANDLES` for all I/O operations on Windows to overcome a hard limit of 2048 open file descriptors in Microsoft's C runtime library. [#5249]
 13. Fix a bug where Berkeley DB error message routines could drop core on the PowerPC and UltraSPARC architectures. [#5331]
 14. Rename `OSTREAMCLASS` to `__DB_OSTREAMCLASS` in `db_cxx.h` to avoid stepping on application name space. [#5402]
 15. Support Linux on the S/390 architecture. [#5608]
 16. Work around a bug in Solaris where the `pthread_cond_wait` call could return because a signal was delivered to the application. [#5640]
 17. Fix build line for loadable libraries to include `-module` to support Mac OS X. [#5664]
 18. Fix a bug in the PPC mutex support for the Mac OS X system. [#5781]
 19. Added support for Java on Mac OS X. A workaround on the Java command line is currently necessary; it is documented. [#5664]
 20. Added support for Tcl on Mac OS X. [#5664]
 21. Update Windows build instructions to cover Visual C++ .NET. [#5684]
 22. AIX configuration changes for building on AIX 4.3.3 and 5 with both standard and Visual Age compilers. [#5779]
 23. Add a new UNIX configuration argument, `--with-mutex=MUTEX`, to allow applications to select a mutex implementation. [#6040]
 24. Changed `libtool` and `configure` so we can now correctly build and install Tcl and Java loadable shared libraries that work on Mac OS X. [#6117]
 25. Fix mutex alignment problems on historic HP-UX releases that could make multiprocess applications fail. [#6250]
 26. Installed static `.a` archives on Mac OS X need to be built with the `ranlib -c` option so linked applications will not see undefined `__db_jump` errors. [#6215]
 27. Upgrade `pthread` and `mmap` support in the `uClibc` library to support Berkeley DB. [#6268]
 28. Fixed error in determining include directories during configuration for `--enable-java`. The error can cause compilation errors on certain systems with newer versions of `gcc`. [#6445]

Berkeley DB 4.1.25 Change Log

Berkeley DB version 4.1.25 is version 4.1.24 with all public patches applied. There were no public interface changes or new features.

Chapter 10. Upgrading Berkeley DB 3.3 applications to Berkeley DB 4.0

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 3.3 release interfaces to the Berkeley DB 4.0 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

db_deadlock

The `-w` option to the `db_deadlock` utility has been deprecated. Applications can get the functionality of the `-w` option by using the `-t` option with an argument of `.100000`.

lock_XXX

The C API for the Berkeley DB Locking subsystem was reworked in the 4.0 release as follows:

Historic functional interface	Berkeley DB 4.X method
<code>lock_detect</code>	<code>DB_ENV->lock_detect()</code>
<code>lock_get</code>	<code>DB_ENV->lock_get()</code>
<code>lock_id</code>	<code>DB_ENV->lock_id()</code>
<code>lock_put</code>	<code>DB_ENV->lock_put()</code>
<code>lock_stat</code>	<code>DB_ENV->lock_stat()</code>
<code>lock_vec</code>	<code>DB_ENV->lock_vec()</code>

Applications calling any of these functions should update their calls to use the enclosing `DB_ENV` handle's method (easily done as the first argument to the existing call is the correct handle to use).

In addition, the `DB_ENV->lock_stat()` call has been changed in the 4.0 release to take a flags argument. To leave their historic behavior unchanged, applications should add a final argument of 0 to any calls made to `DB_ENV->lock_stat()`.

The C++ and Java APIs for the `DbLock::put` (`DbLock.put`) method was reworked in the 4.0 release to make the lock put interface a method of the `DB_ENV` handle rather than the `DbLock` handle. Applications calling the `DbLock::put` or `DbLock.put` method should update their calls to use the enclosing `DB_ENV` handle's method (easily done as the first argument to the existing call is the correct handle to use).

log_XXX

The C API for the Berkeley DB Logging subsystem was reworked in the 4.0 release as follows:

Historic functional interface	Berkeley DB 4.X method
<code>log_archive</code>	<code>DB_ENV->log_archive()</code>

Historic functional interface	Berkeley DB 4.X method
log_file	DB_ENV->log_file()
log_flush	DB_ENV->log_flush()
log_get	DB_ENV->log_cursor()
log_put	DB_ENV->log_put()
log_register	DB_ENV->log_register
log_stat	DB_ENV->log_stat()
log_unregister	DB_ENV->log_unregister

Applications calling any of these functions should update their calls to use the enclosing DB_ENV class handle's method (in all cases other than the log_get call, this is easily done as the first argument to the existing call is the correct handle to use).

Application calls to the historic log_get function must be replaced with the creation of a log file cursor (a DB_LOGC class object), using the DB_ENV->log_cursor() method to retrieve log records and calls to the DB_LOGC->close() method to destroy the cursor. It may also be possible to simplify some applications. In previous releases of Berkeley DB, the DB_CURRENT, DB_NEXT, and DB_PREV flags to the log_get function could not be used by a free-threaded DB_ENV class handle. If their DB_ENV class handle was free-threaded, applications had to create an additional, unique environment handle by separately calling DB_ENV->open(). This is no longer an issue in the log cursor interface, and applications may be able to remove the now unnecessary creation of the additional DB_ENV class object.

Finally, the DB_ENV->log_stat() call has been changed in the 4.0 release to take a flags argument. To leave their historic behavior unchanged, applications should add a final argument of 0 to any calls made to DB_ENV->log_stat().

memp_XXX

The C API for the Berkeley DB Memory Pool subsystem was reworked in the 4.0 release as follows:

Historic functional interface	Berkeley DB 4.X method
memp_register	DB_ENV->memp_register()
memp_stat	DB_ENV->memp_stat()
memp_sync	DB_ENV->memp_sync()
memp_trickle	DB_ENV->memp_trickle()
memp_fopen	DB_ENV->memp_fcreate()
DB_MPOOL_FINFO: ftype	DB_MPOOLFILE->set_ftype()
DB_MPOOL_FINFO: pgcookie	DB_MPOOLFILE->set_pgcookie()
DB_MPOOL_FINFO: fileid	DB_MPOOLFILE->set_fileid()
DB_MPOOL_FINFO: lsn_offset	DB_MPOOLFILE->set_lsn_offset()
DB_MPOOL_FINFO: clear_len	DB_MPOOLFILE->set_clear_len()

Historic functional interface	Berkeley DB 4.X method
memp_fopen	DB_MPOOLFILE->open()
memp_fclose	DB_MPOOLFILE->close()
memp_fput	DB_MPOOLFILE->put()
memp_fset	DB_MPOOLFILE->set
memp_fsync	DB_MPOOLFILE->sync()

Applications calling any of the `memp_register`, `memp_stat`, `memp_sync` or `memp_trickle` functions should update those calls to use the enclosing `DB_ENV` class handle's method (easily done as the first argument to the existing call is the correct `DB_ENV` class handle).

In addition, the `DB_ENV->memp_stat()` call has been changed in the 4.0 release to take a flags argument. To leave their historic behavior unchanged, applications should add a final argument of 0 to any calls made to `DB_ENV->memp_stat()`.

Applications calling the `memp_fopen` function should update those calls as follows: First, acquire a Cache chapter handle using the `DB_ENV->memp_fcreate()` method. Second, if the `DB_MPOOL_FINFO` structure reference passed to the `memp_fopen` function was non-NULL, call the Cache chapter method corresponding to each initialized field in the `DB_MPOOL_FINFO` structure. Third, call the `DB_MPOOLFILE->open()` method method to open the underlying file. If the `DB_MPOOLFILE->open()` method call fails, then `DB_MPOOLFILE->close()` method must be called to destroy the allocated handle.

Applications calling the `memp_fopen`, `memp_fclose`, `memp_fput`, `memp_fset`, or `memp_fsync` functions should update those calls to use the enclosing Cache chapter handle's method. Again, this is easily done as the first argument to the existing call is the correct Cache chapter handle. With one exception, the calling conventions of the old and new interfaces are identical; the one exception is the `DB_MPOOLFILE->close()` method, which requires an additional flag parameter that should be set to 0.

txn_XXX

The C API for the Berkeley DB Transaction subsystem was reworked in the 4.0 release as follows:

Historic functional interface	Berkeley DB 4.X method
txn_abort	DB_TXN->abort()
txn_begin	DB_ENV->txn_begin()
txn_checkpoint	DB_ENV->txn_checkpoint()
txn_commit	DB_TXN->commit()
txn_discard	DB_TXN->discard()
txn_id	DB_TXN->id()
txn_prepare	DB_TXN->prepare()
txn_recover	DB_TXN->recover()

Historic functional interface	Berkeley DB 4.X method
txn_stat	DB_TXN->stat()

Applications calling any of these functions should update their calls to use the enclosing DB_ENV class handle's method (easily done as the first argument to the existing call is the correct handle to use).

As a special case, since applications might potentially have many calls to the txn_abort, txn_begin and txn_commit functions, those functions continue to work unchanged in the Berkeley DB 4.0 release.

In addition, the DB_TXN->stat() call has been changed in the 4.0 release to take a flags argument. To leave their historic behavior unchanged, applications should add a final argument of 0 to any calls made to DB_TXN->stat().

db_env_set_XXX

The db_env_set_region_init function was removed in the 4.0 release and replaced with the DB_REGION_INIT flag to the DB_ENV->set_flags() method. This is an interface change: historically, the db_env_set_region_init function operated on the entire Berkeley DB library, not a single environment. The new method only operates on a single DB_ENV class handle (and any handles created in the scope of that handle). Applications calling the db_env_set_region_init function should update their calls: calls to the historic routine with an argument of 1 (0) are equivalent to calling DB_ENV->set_flags() with the DB_REGION_INIT flag and an argument of 1 (0).

The db_env_set_tas_spins function was removed in the 4.0 release and replaced with the DB_ENV->set_tas_spins method. This is an interface change: historically, the db_env_set_tas_spins function operated on the entire Berkeley DB library, not a single environment. The new method only operates on a single DB_ENV class handle (and any handles created in the scope of that handle). Applications calling the db_env_set_tas_spins function should update their calls: calls to the historic routine are equivalent to calling DB_ENV->set_tas_spins with the same argument. In addition, for consistent behavior, all DB_ENV class handles opened by the application should make the same configuration call, or the value will need to be entered into the environment's DB_CONFIG file.

Also, three of the standard Berkeley DB debugging interfaces changed in the 4.0 release. It is quite unlikely that Berkeley DB applications use these interfaces.

The DB_ENV->set_mutexlocks method was removed in the 4.0 release and replaced with the DB_NO_LOCKING flag to the DB_ENV->set_flags() method. Applications calling the DB_ENV->set_mutexlocks method should update their calls: calls to the historic routine with an argument of 1 (0) are equivalent to calling DB_NO_LOCKING flag and an argument of 1 (0).

The db_env_set_pageyield function was removed in the 4.0 release and replaced with the DB_YIELDCPU flag to the DB_ENV->set_flags() method. This is an interface change: historically, the db_env_set_pageyield function operated on the entire Berkeley DB library, not a single environment. The new method only operates on a single DB_ENV class handle (and any handles created in the scope of that handle). Applications calling the db_env_set_pageyield

function should update their calls: calls to the historic routine with an argument of 1 (0) are equivalent to calling `DB_ENV->set_flags()` with the `DB_YIELDCPU` flag and an argument of 1 (0). In addition, all `DB_ENV` class handles opened by the application will need to make the same call, or the `DB_YIELDCPU` flag will need to be entered into the environment's `DB_CONFIG` file.

The `db_env_set_panicstate` function was removed in the 4.0 release, replaced with the `DB_PANIC_ENVIRONMENT` flags to the `DB_ENV->set_flags()` method. (The `DB_PANIC_ENVIRONMENT` flag will cause an environment to panic, affecting all threads of control using that environment. The `DB_ENV->set_flags()` handle to ignore the current panic state of the environment.) This is an interface change: historically the `db_env_set_panicstate` function operated on the entire Berkeley DB library, not a single environment. Applications calling the `db_env_set_panicstate` function should update their calls, replacing the historic call with a call to `DB_ENV->set_flags()` and the appropriate flag, depending on their usage of the historic interface.

DB_ENV->set_server

The `DB_ENV->set_server()` method has been replaced with the `DB_ENV->set_rpc_server()` method. The `DB_ENV->set_server()` method can be easily converted to the `DB_ENV->set_rpc_server()` method by changing the name, and specifying a `NULL` for the added argument, second in the argument list.

DB_ENV->set_lk_max

The `DB_ENV->set_lk_max` method has been deprecated in favor of the `DB_ENV->set_lk_max_locks()`, `DB_ENV->set_lk_max_lockers()`, and `DB_ENV->set_lk_max_objects()` methods. The `DB_ENV->set_lk_max` method continues to be available, but is no longer documented and is expected to be removed in a future release.

DB_ENV->lock_id_free

A new locker ID related API, the `DB_ENV->lock_id_free()` method, was added to Berkeley DB 4.0 release. Applications using the `DB_ENV->lock_id()` method to allocate locker IDs may want to update their applications to free the locker ID when it is no longer needed.

Java CLASSPATH environment variable

The Berkeley DB Java class files are now packaged as jar files. In the 4.0 release, the `CLASSPATH` environment variable must change to include at least the `db.jar` file. It can optionally include the `dbexamples.jar` file if you want to run the examples. For example, on UNIX:

```
export CLASSPATH="/usr/local/BerkeleyDB.4.8/lib/db.jar: \
/usr/local/BerkeleyDB.4.8/lib/dbexamples.jar"
```

For example, on Windows:

```
set CLASSPATH="D:\db\build_windows\Release\db.jar;
```

```
D:\db\build_windows\Release\dbexamples.jar"
```

For more information on Java configuration, see the [Berkeley DB Installation and Build Guide](#).

C++ ostream objects

In the 4.0 release, the Berkeley DB C++ API has been changed to use the ISO standard C++ API in preference to the older, less portable interfaces, where available. This means the Berkeley DB methods that used to take an ostream object as a parameter now expect a std::ostream. Specifically, the following methods have changed:

```
DbEnv::set_error_stream  
Db::set_error_stream  
Db::verify
```

On many platforms, the old and the new C++ styles are interchangeable; on some platforms (notably Windows systems), they are incompatible. If your code uses these methods and you have trouble with the 4.0 release, you should update code that looks like this:

```
#include <iostream.h>  
#include <db_cxx.h>  
  
void foo(Db db) {  
    db.set_error_stream(&cerr);  
}
```

to look like this:

```
#include <iostream>  
#include <db_cxx.h>  
  
using std::cerr;  
  
void foo(Db db) {  
    db.set_error_stream(&cerr);  
}
```

application-specific recovery

If you have created your own logging and recovery routines, you may need to upgrade them to the Berkeley DB 4.0 release.

First, you should regenerate your logging, print, read and the other automatically generated routines, using the dist/gen_rec.awk tool included in the Berkeley DB distribution.

Next, compare the template file code generated by the gen_rec.awk tool against the code generated by the last release in which you built a template file. Any changes in the templates should be incorporated into the recovery routines you have written.

Third, if your recovery functions refer to DB_TXN_FORWARD_ROLL (that is, your code checks for that particular operation code), you should replace it with DB_REDO(op) which compares

the operation code to both DB_TXN_FORWARD_ROLL and DB_TXN_APPLY. (DB_TXN_APPLY is a potential value for the operation code as of the 4.0 release.)

Finally, if you have created your own logging and recovery routines, we recommend you contact us and ask us to review those routines for you.

Upgrade Requirements

The log file format changed in the Berkeley DB 4.0 release. No database formats changed in the Berkeley DB 4.0 release.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

4.0.14 Change Log

Major New Features:

1. Group commit. [#42]
2. Single-master replication. [#44]
3. Support for VxWorks AE; Vxworks support certified by WindRiver Systems Inc. [#4401]

General Environment Changes:

1. The db_env_set_pageyield interface has been replaced by a new flag (DB_YIELDCPU) for the DB_ENV->set_flags interface.
2. The db_env_set_panicstate interface has been replaced by a new flag (DB_PANIC_STATE) for the DB_ENV->set_flags interface.
3. The db_env_set_region_init interface has been replaced by a new flag (DB_REGION_INIT) for the DB_ENV->set_flags interface.
4. The db_env_set_tas_spins interface has been replaced by the DB_ENV->set_tas_spins method.
5. The DB_ENV->set_mutexlocks interface has been replaced by a new flag (DB_NOLOCKING) for the DB_ENV->set_flags interface.
6. Fix a bug where input values from the DB_CONFIG file could overflow.
7. The C API lock, log, memory pool and transaction interfaces have been converted to method based interfaces; see the Upgrade documentation for specific details. [#920]
8. Fix a bug in which some DB_ENV configuration information could be lost by a failed DB_ENV->open command. [#4608]
9. Fix a bug where Berkeley DB could fail if the application attempted to allocate new database pages while the system was unable to write new log file buffers. [#4928]

General Access Method Changes:

1. Add a new flag (DB_GET_BOTH_RANGE) that adds support for range searches within sorted duplicate data sets. [#3378]
2. Fix a bug in which the DB->get or DB->pget methods, when used with secondary indices, could incorrectly leave an internally-created database cursor open. [#4465]
3. The DB->set_alloc method can no longer be called when the database is part of a database environment. [#4599]

Btree Access Method Changes:

1. Fix a bug where a lock could be leaked when a thread calling DB->stat on a Btree database was selected to resolve a deadlock. [#4509]

Hash Access Method Changes:

1. Fix a bug where bulk return using the MULTIPLE_KEY flag on a Hash database would only return entries from a single bucket. [#4313]

Queue Access Method Changes:

1. Delete extent files whenever the leading record is deleted, instead of only when a DB_CONSUME operation was performed. [#4307]

Recno Access Method Changes:

1. Fix a bug where the delete of a record in a Recno database could leak a lock in non-transactional applications. [#4351]
2. Fix a bug where the DB_THREAD flag combined with a backing source file could cause an infinite loop. [#4581]

C++ API Changes:**Java API Changes:**

1. Added implementation of DbEnv.lock_vec for Java. [#4094] Added some minimal protection so that the same Java Dbt cannot be used twice in the same API call, this will often catch multithreading programming errors with Dbts. [#4094]
2. Fix a bug in which a Db.put call with the Db.DB_APPEND would fail to correctly return the newly put record's record number. [#4527]
3. Fixed problems occurring in multithreaded java apps that use callbacks. [#4467]

Tcl API Changes:

1. Fix a bug in which large integers could be handled incorrectly by the Tcl interface on 64-bit machines. [#4371]

RPC Client/Server Changes:

1. The DB_ENV->set_server interface has been removed.

XA Resource Manager Changes:

Locking Subsystem Changes:

1. The C++ (Java) API DbLock::put (DbLock.put) method has been changed to be a method off the DbEnv handle rather than the DbLock handle.
2. Locker IDs may now wrap-around. [#864]
3. Explicitly allocated locker IDs must now be freed. [#864]
4. Add per-environment, per-lock and per-transaction interfaces to support timeout based lock requests and "deadlock" detection. [#1855]
5. Add support for interrupting a waiting locker. [#1976]
6. Implemented DbEnv.lock_vec for Java. [#4094]

Logging Subsystem Changes:

1. Fix a bug where the size of a log file could not be set to the default value. [#4567]
2. Fix a bug where specifying a non-default log file size could cause other processes to be unable to join the environment and read its log files. [#4567]
3. Fix a bug where Berkeley DB could keep open file descriptors to log files returned by the DB_ENV->log_archive method (or the db_archive utility), making it impossible to move or remove them on Windows systems. [#3969]
4. Replace the log_get interface with a cursor into the log file. [#0043]

Memory Pool Subsystem Changes:

1. Add the DB_ODDFILESIZE flag to the DB_MPOOLFILE->open method supporting files not a multiple of the underlying page size in length.
2. Convert memp_XXX functional interfaces to a set of methods, either base methods off the DB_ENV handle or methods off of a DB_MPOOLFILE handle. [#920]
3. Add the DB_ODDFILESIZE flag to the DB_MPOOLFILE->open method supporting files not a multiple of the underlying page size in length.
4. Fix a bug where threads of control could deadlock opening a database environment with multiple memory pool caches. [#4696]
5. Fix a bug where the space needed for per-file memory pool statistics was incorrectly calculated. [#4772]

Transaction Subsystem Changes:

1. Transaction IDs may now wrap-around. [#864]
2. Release read locks before performing logging operations at commit. [#4219]

Utility Changes:

1. Fix a bug in which the db_dump utility would incorrectly attach to transaction, locking, or logging regions when salvaging, and thus could not be used to salvage databases in environments where these regions were present. [#4305]
2. Fix a bug in which the DB salvager could produce incorrectly formatted output for certain classes of corrupt database. [#4305]
3. Fix a bug in which the DB salvager could incorrectly salvage files containing multiple databases. [#4305]
4. Fix a bug where unprintable characters in subdatabase names could cause a dump of a database that could not then be loaded. [#4688]
5. Increase the size of the cache created by the db_stat and db_verify utilities to avoid failure on large databases. [#4688] [#4787]
6. Fix a bug in which a database verification performed with the DB_ORDERCHKONLY flag could fail incorrectly. [#4757]
7. Fix a bug which caused db_stat to display incorrect information about GB size caches. [#4812]

Database or Log File On-Disk Format Changes:

1. The on-disk log format changed.

Configuration, Documentation, Portability and Build Changes:

1. Fix a bug where Win9X systems region names could collide.
2. Fix a bug where configuring Berkeley DB to build the C++ API without also configuring for a shared library build would fail to build the C++ library. [#4343]
3. Change Berkeley DB installation to not strip binaries if --enable-debug was specified as a configuration option. [#4318]
4. Add the -pthread flag to AIX, FreeBSD and OSF/1 library loads. [#4350]
5. Fix a bug where the Berkeley DB 1.85 compatibility API failed to load in the 3.3.11 release. [#4368]
6. Port the Berkeley DB utility programs to the VxWorks environment. [#4378]
7. Made change to configuration so that dynamic libraries link correctly when C++ is used on AIX. [#4381]

8. Fix a variety of problems that prevented the Berkeley DB source tree from building on systems without ANSI C compiler support (for example, SunOS 4.X). [#4398]
9. Added missing DbMultiple*Iterator Java files to Makefile.in. [#4404]
10. Fix a bug that could prevent the db_dump185 utility from dumping Berkeley DB version 1.86 hash databases. [#4418]
11. Reduce the number of calls setting the errno value, to improve performance on Windows/NT in MT environments. [#4432]
12. Fix for Darwin (and probably some other) OS's that were getting 'yes' or other garbage in generated makefiles in place of a shared library name. [#4453]
13. C++: Remove inlining for constructor of tmpString internal class. This fixes warnings on Solaris profiling builds. [#4473]
14. DB now restarts system calls that are interrupted by signals. [#4480]
15. Fixed warnings for compiling Java native code on Solaris and OSF/1. [#4571]
16. Added better configuration for Java on Tru64 (OSF/1), Solaris,
17. Java files are now built as jar files. Berkeley DB classes are put into db.jar (which is an installed file on UNIX) and examples are put into dbexamples.jar. The classes directory is now a subdirectory of the build directory, rather than in java/classes. [#4575]
18. Support Cygwin installation process. [#4611]
19. Correct the Java secondary_key_create method signature. [#4777]
20. Export additional Berkeley DB interfaces on Windows to support application-specific logging and recovery. [#4827]
21. Always complain when using version 2.96 of the gcc compiler. [#4878]
22. Add compile and load-time flags to configure for threads on UnixWare and OpenUNIX. [#4552] [#4950]

Chapter 11. Upgrading Berkeley DB 3.2 applications to Berkeley DB 3.3

introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 3.2 release interfaces to the Berkeley DB 3.3 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

DB_ENV->set_server

The DB_ENV->set_server method has been deprecated and replaced with the DB_ENV->set_rpc_server() method. The DB_ENV->set_server method will be removed in a future release, and so applications using it should convert. The DB_ENV->set_server method can be easily converted to the DB_ENV->set_rpc_server() method by changing the name, and specifying a NULL for the added argument, second in the argument list.

DB->get_type

The DB->get_type() method can return an error in the Berkeley DB 3.3 release, and so requires an interface change. C and C++ applications calling DB->get_type() should be changed to treat the method's return as an error code, and to pass an additional second argument of type DBTYPE * to the method. The additional argument is used as a memory location in which to store the requested information.

DB->get_byteswapped

The DB->get_byteswapped() method can return an error in the Berkeley DB 3.3 release, and so requires an interface change. C and C++ applications calling DB->get_byteswapped() should be changed to treat the method's return as an error code, and to pass an additional second argument of type int * to the method. The additional argument is used as a memory location in which to store the requested information.

DB->set_malloc, DB->set_realloc

There are two new methods in the Berkeley DB 3.3 release: DB_ENV->set_alloc(). These functions allow applications to specify a set of allocation functions for the Berkeley DB library to use when allocating memory to be owned by the application and when freeing memory that was originally allocated by the application.

The new methods affect or replace the following historic methods:

DB->set_malloc

The DB->set_malloc method has been replaced in its entirety. Applications using this method should replace the call with a call to DB->set_alloc().

DB->set_realloc

The DB->set_realloc method has been replaced in its entirety. Applications using this method should replace the call with a call to DB->set_alloc().

DB->stat() method

has been replaced. Applications using this method should do as follows: if the argument is NULL, it should simply be removed. If non-NULL, it should be replaced with a call to DB->set_alloc().

lock_stat

The historic **db_malloc** argument to the lock_stat function has been replaced. Applications using this function should do as follows: if the argument is NULL, it should simply be removed. If non-NULL, it should be replaced with a call to DB_ENV->set_alloc().

log_archive

The historic **db_malloc** argument to the log_archive function has been replaced. Applications using this function should do as follows: if the argument is NULL, it should simply be removed. If non-NULL, it should be replaced with a call to DB_ENV->set_alloc().

log_stat

The historic **db_malloc** argument to the log_stat function has been replaced. Applications using this function should do as follows: if the argument is NULL, it should simply be removed. If non-NULL, it should be replaced with a call to DB_ENV->set_alloc().

memp_stat

The historic **db_malloc** argument to the memp_stat function has been replaced. Applications using this function should do as follows: if the argument is NULL, it should simply be removed. If non-NULL, it should be replaced with a call to DB_ENV->set_alloc().

txn_stat

The historic **db_malloc** argument to the txn_stat function has been replaced. Applications using this function should do as follows: if the argument is NULL, it should simply be removed. If non-NULL, it should be replaced with a call to DB_ENV->set_alloc().

One potential incompatibility for historic applications is that the allocation functions for a database environment must now be set before the environment is opened. Historically, Berkeley DB applications could open the environment first, and subsequently call the DB->set_malloc and DB->set_realloc methods; that use is no longer supported.

DB_LOCK_CONFLICT

The DB_LOCK_CONFLICT flag has been removed from the lock_detect function. Applications specifying the DB_LOCK_CONFLICT flag should simply replace it with a flags argument of 0.

memp_fget, EIO

Previous releases of Berkeley DB returned the system error EIO when the memp_fget function was called to retrieve a page, the page did not exist, and the DB_MPOOL_CREATE flag was not set. In the 3.3 release, the error DB_PAGE_NOTFOUND is returned instead, to allow applications to distinguish between recoverable and non-recoverable errors. Applications calling the

memp_fget function and checking for a return of EIO should check for DB_PAGE_NOTFOUND instead.

Previous releases of Berkeley DB treated filesystem I/O failure (the most common of which the filesystem running out of space), as a fatal error, returning DB_RUNRECOVERY. When a filesystem failure happens in the 3.3 release Berkeley DB returns the underlying system error (usually EIO), but can continue to run. Applications should abort any enclosing transaction when a recoverable system error occurs in order to recover from the error.

txn_prepare

An additional argument has been added to the txn_prepare function. If your application calls txn_prepare (that is, is performing two-phase commit using Berkeley DB as a local resource manager), see the section titled *Distributed Transactions* in versions of this book that existed prior to release 4.8.

--enable-dynamic, --enable-shared

In previous releases, Berkeley DB required separate configuration and builds to create both static and shared libraries. This has changed in the 3.3 release, and Berkeley DB now builds and installs both shared and static versions of the Berkeley DB libraries by default. This change was based on Berkeley DB upgrading to release 1.4 of the GNU Project's Libtool distribution. For this reason, Berkeley DB no longer supports the previous --enable-dynamic and --enable-shared configuration options. Instead, as Berkeley DB now builds both static and shared libraries by default, the useful options are Libtool's --disable-shared and --disable-static options.

--disable-bigfile

In previous releases, Berkeley DB UNIX used the --disable-bigfile configuration option for systems that could not, for whatever reason, include large file support in a particular Berkeley DB configuration. However, large file support has been integrated into the autoconf configuration tool as of version 2.50. For that reason, Berkeley DB configuration no longer supports --disable-bigfile, the autoconf standard --disable-largefile should be used instead.

Upgrade Requirements

No database formats or log file formats changed in the Berkeley DB 3.3 release.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Chapter 12. Upgrading Berkeley DB 3.1 applications to Berkeley DB 3.2

introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 3.1 release interfaces to the Berkeley DB 3.2 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

DB_ENV->set_flags

A new method has been added to the Berkeley DB environment handle, DB_ENV->set_flags(). This method currently takes three flags: DB_CDB_ALLDB, DB_NOMMAP, and DB_TXN_NOSYNC. The first of these flags, DB_CDB_ALLDB, provides new functionality, allowing Berkeley DB Concurrent Data Store applications to do locking across multiple databases.

The other two flags, DB_NOMMAP and DB_TXN_NOSYNC, were specified to the DB_ENV->open() method in previous releases. In the 3.2 release, they have been moved to the DB_ENV->set_flags() method because this allows the database environment's value to be toggled during the life of the application as well as because it is a more appropriate place for them. Applications specifying either the DB_NOMMAP or DB_TXN_NOSYNC flags to the DB_ENV->open() method should replace those flags with calls to the DB_ENV->set_flags() method.

DB callback functions, app_private field

In the Berkeley DB 3.2 release, four application callback functions (the callback functions set by DB->set_bt_compare(), DB->set_bt_prefix(), DB->set_dup_compare() and DB->set_h_hash()) were modified to take a reference to a DB object as their first argument. This change allows the Berkeley DB Java API to reasonably support these interfaces. There is currently no need for the callback functions to do anything with this additional argument.

C and C++ applications that specify their own Btree key comparison, Btree prefix comparison, duplicate data item comparison or Hash functions should modify these functions to take a reference to a DB structure as their first argument. No further change is required.

The app_private field of the DBT structure (accessible only from the Berkeley DB C API) has been removed in the 3.2 release. It was replaced with app_private fields in the DB_ENV handles. Applications using this field will have to convert to using one of the replacement fields.

Logically renumbering records

In the Berkeley DB 3.2 release, cursor adjustment semantics changed for Recno databases with mutable record numbers. Before the 3.2 release, cursors were adjusted to point to the previous or next record at the time the record to which the cursor referred was deleted. This could lead to unexpected behaviors. For example, two cursors referring to sequential records that were both deleted would lose their relationship to each other and would refer to the same position in the database instead of their original sequential relationship. There were

also command sequences that would have unexpected results. For example, DB_AFTER and DB_BEFORE cursor put operations, using a cursor previously used to delete an item, would perform the put relative to the cursor's adjusted position and not its original position.

In the Berkeley DB 3.2 release, cursors maintain their position in the tree regardless of deletion operations using the cursor. Applications that perform database operations, using cursors previously used to delete entries in Recno databases with mutable record numbers, should be evaluated to ensure that the new semantics do not cause application failure.

DB_INCOMPLETE

There are a number of functions that flush pages from the Berkeley DB shared memory buffer pool to disk. Most of those functions can potentially fail because a page that needs to be flushed is not currently available. However, this is not a hard failure and is rarely cause for concern. In the Berkeley DB 3.2 release, the C++ API (if that API is configured to throw exceptions) and the Java API have been changed so that this failure does not throw an exception, but rather returns a non-zero error code of DB_INCOMPLETE.

The following C++ methods will return DB_INCOMPLETE rather than throw an exception: Db::close, Db::sync, DbEnv::memp_sync, DbEnv::txn_checkpoint, and DbMpoolFile::memp_fsync.

The following Java methods are now declared "public int" rather than "public void", and will return Db.DB_INCOMPLETE rather than throw an exception: Db.close(), Db.sync(), and DbEnv.checkpoint().

It is likely that the only change required by any application will be those currently checking for a DB_INCOMPLETE return that has been encapsulated in an exception.

DB_ENV->set_tx_recover

The `info` parameter of the function passed to DB_ENV->set_tx_recover is no longer needed. If your application calls DB_ENV->set_tx_recover, find the callback function referred to by that call and remove the `info` parameter.

In addition, the called function no longer needs to handle Berkeley DB log records, Berkeley DB will handle them internally as well as call the application-specified function. Any handling of Berkeley DB log records in the application's callback function may be removed.

In addition, the callback function will no longer be called with the DB_TXN_FORWARD_ROLL flag specified unless the transaction enclosing the operation successfully committed.

DB_ENV->set_mutexlocks

Previous Berkeley DB releases included the `db_env_set_mutexlocks` function, intended for debugging, that allows applications to always obtain requested mutual exclusion mutexes without regard for their availability. This function has been replaced with `dbenv_set_mutexlocks`, which provides the same functionality on a per-database environment basis. Applications using the old function should be updated to use the new one.

Java and C++ object reuse

In previous releases of Berkeley DB, Java DbEnv and Db objects, and C++ DbEnv and Db objects could be reused after they were closed, by calling open on them again. This is no longer permitted, and these objects no longer allow any operations after a close. Applications reusing these objects should be modified to create new objects instead.

Java java.io.FileNotFoundException

The Java DbEnv.remove, Db.remove and Db.rename methods now throw java.io.FileNotFoundException in the case where the named file does not exist. Applications should be modified to catch this exception where appropriate.

db_dump

In previous releases of Berkeley DB, the db_dump utility dumped Recno access method database keys as numeric strings. For consistency, the db_dump utility has been changed in the 3.2 release to dump record numbers as hex pairs when the data items are being dumped as hex pairs. (See the **-k** and **-p** options to the db_dump utility for more information.) Any applications or scripts post-processing the output of the db_dump utility for Recno databases under these conditions may require modification.

Upgrade Requirements

Log file formats and the Queue Access Method database formats changed in the Berkeley DB 3.2 release. (The on-disk Queue format changed from version 2 to version 3.) Until the underlying databases are upgraded, the DB_OLD_VERSION error.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Chapter 13. Upgrading Berkeley DB 3.0 applications to Berkeley DB 3.1

introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 3.0 release interfaces to the Berkeley DB 3.1 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

DB_ENV->open, DB_ENV->remove

In the Berkeley DB 3.1 release, the **config** argument to the DB_ENV->open() and DB_ENV->remove() methods has been removed, replaced by additional methods on the DB_ENV handle. If your application calls DB_ENV->open() or DB_ENV->remove() with a NULL **config** argument, find those functions and remove the config argument from the call. If your application has non-NULL **config** argument, the strings values in that argument are replaced with calls to DB_ENV methods as follows:

Previous config string	Berkeley DB 3.1 version method
DB_DATA_DIR	DB_ENV->set_data_dir()
DB_LOG_DIR	DB_ENV->set_log_dir()
DB_TMP_DIR	DB_ENV->set_tmp_dir()

DB_ENV->set_tx_recover

The redo parameter of the function passed to DB_ENV->set_tx_recover used to be an integer set to any one of a number of #defined values. In the 3.1 release of Berkeley DB, the redo parameter has been replaced by the op parameter which is an enumerated type of type db_recops.

If your application calls DB_ENV->set_tx_recover, then find the function referred to by the call. Replace the flag values in that function as follows:

Previous flag	Berkeley DB 3.1 version flag
TXN_BACKWARD_ROLL	DB_TXN_BACKWARD_ROLL
TXN_FORWARD_ROLL	DB_TXN_FORWARD_ROLL
TXN_OPENFILES	DB_TXN_OPENFILES
TXN_REDO	DB_TXN_FORWARD_ROLL
TXN_UNDO	DB_TXN_ABORT

DB_ENV->set_feedback, DB->set_feedback

Starting with the 3.1 release of Berkeley DB, the DB_ENV->set_feedback() and DB->set_feedback() methods may return an error value, that is, they are no longer declared as

returning no value, instead they return an int or throw an exception as appropriate when an error occurs.

If your application calls these functions, you may want to check for a possible error on return.

DB_ENV->set_paniccall, DB->set_paniccall

Starting with the 3.1 release of Berkeley DB, the DB_ENV->set_paniccall and DB->set_paniccall methods may return an error value, that is, they are no longer declared as returning no value, instead they return an int or throw an exception as appropriate when an error occurs.

If your application calls these functions, you may want to check for a possible error on return.

DB->put

For the Queue and Recno access methods, when the DB_APPEND flag is specified to the DB->put() method, the allocated record number is returned to the application in the **key** DBT argument. In previous releases of Berkeley DB, this DBT structure did not follow the usual DBT conventions. For example, it was not possible to cause Berkeley DB to allocate space for the returned record number. Rather, it was always assumed that the **data** field of the **key** structure referred to memory that could be used as storage for a db_recno_t type.

As of the Berkeley DB 3.1.0 release, the **key** structure behaves as described in the DBT C++/Java class or C structure documentation.

Applications which are using the DB_APPEND flag for Queue and Recno access method databases will require a change to upgrade to the Berkeley DB 3.1 releases. The simplest change is likely to be to add the DB_DBT_USERMEM flag to the **key** structure. For example, code that appears as follows:

```
DBT key;
db_recno_t recno;

memset(&key, 0, sizeof(DBT));
key.data = &recno;
key.size = sizeof(recno);
DB->put(DB, NULL, &key, &data, DB_APPEND);
printf("new record number is %lu\n", (u_long)recno);
```

would be changed to:

```
DBT key;
db_recno_t recno;

memset(&key, 0, sizeof(DBT));
key.data = &recno;
key.ulen = sizeof(recno);
key.flags = DB_DBT_USERMEM;
DB->put(DB, NULL, &key, &data, DB_APPEND);
printf("new record number is %lu\n", (u_long)recno);
```

Note that the **ulen** field is now set as well as the flag value. An alternative change would be:

```
DBT key;
db_recno_t recno;

memset(&key, 0, sizeof(DBT));
DB->put(DB, NULL, &key, &data, DB_APPEND);
recno = *(db_recno_t *)key->data;
printf("new record number is %lu\n", (u_long)recno);
```

identical duplicate data items

In previous releases of Berkeley DB, it was not an error to store identical duplicate data items, or, for those that just like the way it sounds, duplicate duplicates. However, there were implementation bugs where storing duplicate duplicates could cause database corruption.

In this release, applications may store identical duplicate data items as long as the data items are unsorted. It is an error to attempt to store identical duplicate data items when duplicates are being stored in a sorted order. This restriction is expected to be lifted in a future release. See Duplicate data items for more information.

DB->stat

For Btree database statistics, the DB->stat() method field **bt_nrecs** has been removed, replaced by two fields: **bt_nkeys** and **bt_ndata**. The **bt_nkeys** field returns a count of the unique keys in the database. The **bt_ndata** field returns a count of the key/data pairs in the database. Neither exactly matches the previous value of the **bt_nrecs** field, which returned a count of keys in the database, but, in the case of Btree databases, could overcount as it sometimes counted duplicate data items as unique keys. The application should be searched for any uses of the **bt_nrecs** field and the field should be changed to be either **bt_nkeys** or **bt_ndata**, whichever is more appropriate.

For Hash database statistics, the DB->stat() method field **hash_nrecs** has been removed, replaced by two fields: **hash_nkeys** and **hash_ndata**. The **hash_nkeys** field returns a count of the unique keys in the database. The **hash_ndata** field returns a count of the key/data pairs in the database. The new **hash_nkeys** field exactly matches the previous value of the **hash_nrecs** field. The application should be searched for any uses of the **hash_nrecs** field, and the field should be changed to be **hash_nkeys**.

For Queue database statistics, the DB->stat() method field **qs_nrecs** has been removed, replaced by two fields: **qs_nkeys** and **qs_ndata**. The **qs_nkeys** field returns a count of the unique keys in the database. The **qs_ndata** field returns a count of the key/data pairs in the database. The new **qs_nkeys** field exactly matches the previous value of the **qs_nrecs** field. The application should be searched for any uses of the **qs_nrecs** field, and the field should be changed to be **qs_nkeys**.

DB_SYSTEM_MEM

Using the DB_SYSTEM_MEM option on UNIX systems now requires the specification of a base system memory segment ID, using the DB_ENV->set_shm_key() method. Any valid segment ID may be specified, for example, one returned by the UNIX ftok(3) function.

log_register

The arguments to the `log_register` and `log_unregister` interfaces have changed. Instead of returning (and passing in) a logging file ID, a reference to the DB structure being registered (or unregistered) is passed. The application should be searched for any occurrences of `log_register` and `log_unregister`. For each one, change the arguments to be a reference to the DB structure being registered or unregistered.

memp_register

An additional argument has been added to the `pgin` and `pgout` functions provided to the `memp_register` function. The application should be searched for any occurrences of `memp_register`. For each one, if `pgin` or `pgout` functions are specified, the `pgin` and `pgout` functions should be modified to take an initial argument of a `DB_ENV *`. This argument is intended to support better error reporting for applications, and may be entirely ignored by the `pgin` and `pgout` functions themselves.

txn_checkpoint

An additional argument has been added to the `txn_checkpoint` function.

The application should be searched for any occurrences of `txn_checkpoint`. For each one, an argument of 0 should be appended to the current arguments.

environment configuration

A set of `DB_ENV` configuration methods which were not environment specific, but which instead affected the entire application space, have been removed from the `DB_ENV` object and replaced by static functions. The following table lists the `DB_ENV` methods previously available to applications and the static functions that should now be used instead.

DB_ENV method	Berkeley DB 3.1 function
<code>DB_ENV->set_func_close</code>	<code>db_env_set_func_close</code>
<code>DB_ENV->set_func_dirfree</code>	<code>db_env_set_func_dirfree</code>
<code>DB_ENV->set_func_dirlist</code>	<code>db_env_set_func_dirlist</code>
<code>DB_ENV->set_func_exists</code>	<code>db_env_set_func_exists</code>
<code>DB_ENV->set_func_free</code>	<code>db_env_set_func_free</code>
<code>DB_ENV->set_func_fsync</code>	<code>db_env_set_func_fsync</code>
<code>DB_ENV->set_func_ioinfo</code>	<code>db_env_set_func_ioinfo</code>
<code>DB_ENV->set_func_malloc</code>	<code>db_env_set_func_malloc</code>
<code>DB_ENV->set_func_map</code>	<code>dbenv_set_func_map</code>
<code>DB_ENV->set_func_open</code>	<code>db_env_set_func_open</code>
<code>DB_ENV->set_func_read</code>	<code>db_env_set_func_read</code>
<code>DB_ENV->set_func_realloc</code>	<code>db_env_set_func_realloc</code>

DB_ENV method	Berkeley DB 3.1 function
DB_ENV->set_func_rename	db_env_set_func_rename
DB_ENV->set_func_seek	db_env_set_func_seek
DB_ENV->set_func_sleep	db_env_set_func_sleep
DB_ENV->set_func_unlink	db_env_set_func_unlink
DB_ENV->set_func_unmap	dbenv_set_func_unmap
DB_ENV->set_func_write	db_env_set_func_write
DB_ENV->set_func_yield	db_env_set_func_yield
DB_ENV->set_pageyield	dbenv_set_pageyield
DB_ENV->set_region_init	dbenv_set_region_init
DB_ENV->set_mutexlocks	dbenv_set_mutexlocks
DB_ENV->set_tas_spins	dbenv_set_tas_spins

Tcl API

The Berkeley DB Tcl API has been modified so that the **-mpool** option to the **berkdb env** command is now the default behavior. The Tcl API has also been modified so that the **-txn** option to the **berkdb env** command implies the **-lock** and **-log** options. Tcl scripts should be updated to remove the **-mpool**, **-lock** and **-log** options.

The Berkeley DB Tcl API has been modified to follow the Tcl standard rules for integer conversion, for example, if the first two characters of a record number are "0x", the record number is expected to be in hexadecimal form.

DB_TMP_DIR

This change only affects Win/32 applications that create in-memory databases.

On Win/32 platforms an additional test has been added when searching for the appropriate directory in which to create the temporary files that are used to back in-memory databases. Berkeley DB now uses any return value from the GetTempPath interface as the temporary file directory name before resorting to the static list of compiled-in pathnames.

If the system registry does not return the same directory as Berkeley DB has been using previously, this change could cause temporary backing files to move to a new directory when applications are upgraded to the 3.1 release. In extreme cases, this could create (or fix) security problems if the file protection modes for the system registry directory are different from those on the directory previously used by Berkeley DB.

log file pre-allocation

This change only affects Win/32 applications.

On Win/32 platforms Berkeley DB no longer pre-allocates log files. The problem was a noticeable performance spike as each log file was created. To turn this feature back on,

search for the flag DB_OSO_LOG in the source file log/log_put.c and make the change described there, or contact us for assistance.

Upgrade Requirements

Log file formats and the Btree, Queue, Recno and Hash Access Method database formats changed in the Berkeley DB 3.1 release. (The on-disk Btree/Recno format changed from version 7 to version 8. The on-disk Hash format changed from version 6 to version 7. The on-disk Queue format changed from version 1 to version 2.) Until the underlying databases are upgraded, the DB->open() method will return a DB_OLD_VERSION error.

An additional flag, DB_DUPSORT, has been added to the DB->upgrade() method for this upgrade. Please review the DB->upgrade() documentation for further information.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Chapter 14. Upgrading Berkeley DB 2.X applications to Berkeley DB 3.0

introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 2.X release interfaces to the Berkeley DB 3.0 release interfaces. This information does not describe how to upgrade Berkeley DB 1.85 release applications.

environment open/close/unlink

The hardest part of upgrading your application from a 2.X code base to the 3.0 release is translating the Berkeley DB environment open, close and remove calls.

There were two logical changes in this part of the Berkeley DB interface. First, in Berkeley DB 3.0, there are no longer separate structures that represent each subsystem (for example, DB_LOCKTAB or DB_TXNMGR) and an overall DB_ENV environment structure. Instead there is only the DB_ENV references should be passed around by your application instead of passing around DB_LOCKTAB or DB_TXNMGR references. This is likely to be a simple change for most applications as few applications use the lock_XXX, log_XXX, memp_XXX or txn_XXX interfaces to create Berkeley DB environments.

The second change is that there are no longer separate open, close, and unlink interfaces to the Berkeley DB subsystems. For example, in previous releases, it was possible to open a lock subsystem either using db_appinit or using the lock_open call. In the 3.0 release the XXX_open interfaces to the subsystems have been removed, and subsystems must now be opened using the 3.0 replacement for the db_appinit call.

To upgrade your application, first find each place your application opens, closes and/or removes a Berkeley DB environment. This will be code of the form:

```
db_appinit, db_appexit
lock_open, lock_close, lock_unlink
log_open, log_close, log_unlink
memp_open, memp_close, memp_unlink
txn_open, txn_close, txn_unlink
```

Each of these groups of calls should be replaced with calls to db_env_create(), DB_ENV->open(), DB_ENV->close(), and DB_ENV->remove().

The db_env_create() call and the call to the DB_ENV->open() method replace the db_appinit, lock_open, log_open, memp_open and txn_open calls. The DB_ENV->close() method replaces the db_appexit, lock_close, log_close, memp_close and txn_close calls. The DB_ENV->remove() call replaces the lock_unlink, log_unlink, memp_unlink and txn_unlink calls.

Here's an example creating a Berkeley DB environment using the 2.X interface:

```
/*
 * db_init --
 * Initialize the environment.
```

```

*/
DB_ENV *
db_init(home)
char *home;
{
    DB_ENV *dbenv;

    if ((dbenv = (DB_ENV *)calloc(sizeof(DB_ENV), 1)) == NULL)
        return (errno);

    if ((errno = db_appinit(home, NULL, dbenv,
        DB_INIT_LOCK | DB_INIT_LOG | DB_INIT_MPOOL | DB_INIT_TXN |
        DB_USE_ENVIRON)) == 0)
        return (dbenv);

    free(dbenv);
    return (NULL);
}

```

In the Berkeley DB 3.0 release, this code would be written as:

```

/*
 * db_init --
 * Initialize the environment.
 */
int
db_init(home, dbenvp)
char *home;
DB_ENV **dbenvp;
{
    int ret;
    DB_ENV *dbenv;

    if ((ret = db_env_create(&dbenv, 0)) != 0)
        return (ret);

    if ((ret = dbenv->open(dbenv, home, NULL,
        DB_INIT_LOCK | DB_INIT_LOG | DB_INIT_MPOOL | DB_INIT_TXN |
        DB_USE_ENVIRON, 0)) == 0) {
        *dbenvp = dbenv;
        return (0);
    }

    (void)dbenv->close(dbenv, 0);
    return (ret);
}

```

As you can see, the arguments to `db_appinit` and to `DB_ENV->open()` are largely the same. There is some minor re-organization: the mapping is that arguments #1, 2, 3, and 4 to `db_appinit` become arguments #2, 3, 1 and 4 to `DB_ENV->open()`. There is one additional

argument to `DB_ENV->open()`, argument #5. For backward compatibility with the 2.X Berkeley DB releases, simply set that argument to 0.

It is only slightly more complex to translate calls to `XXX_open` to the `DB_ENV->open()` method. Here's an example of creating a lock region using the 2.X interface:

```
lock_open(dir, DB_CREATE, 0664, dbenv, &regionp);
```

In the Berkeley DB 3.0 release, this code would be written as:

```
if ((ret = db_env_create(&dbenv, 0)) != 0)
    return (ret);

if ((ret = dbenv->open(dbenv,
    dir, NULL, DB_CREATE | DB_INIT_LOCK, 0664)) == 0) {
    *dbenvp = dbenv;
    return (0);
}
```

Note that in this example, you no longer need the `DB_LOCKTAB` structure reference that was required in Berkeley DB 2.X releases.

The final issue with upgrading the `db_appinit` call is the `DB_MPOOL_PRIVATE` option previously provided for the `db_appinit` call. If your application is using this flag, it should almost certainly use the new `DB_PRIVATE` flag to the `DB_ENV->open()` method. Regardless, you should carefully consider this change before converting to use the `DB_PRIVATE` flag.

Translating `db_appexit` or `XXX_close` calls to `DB_ENV->close()` is equally simple. Instead of taking a reference to a per-subsystem structure such as `DB_LOCKTAB` or `DB_TXNMGR`, all calls take a reference to a `DB_ENV` structure. The calling sequence is otherwise unchanged. Note that as the application no longer allocates the memory for the `DB_ENV` structure, application code to discard it after the call to `db_appexit()` is no longer needed.

Translating `XXX_unlink` calls to `DB_ENV->remove()` is slightly more complex. As with `DB_ENV->close()`, the call takes a reference to a `DB_ENV` structure instead of a per-subsystem structure. The calling sequence is slightly different, however. Here is an example of removing a lock region using the 2.X interface:

```
DB_ENV *dbenv;

ret = lock_unlink(dir, 1, dbenv);
```

In the Berkeley DB 3.0 release, this code fragment would be written as:

```
DB_ENV *dbenv;

ret = dbenv->remove(dbenv, dir, NULL, DB_FORCE);
```

The additional argument to the `DB_ENV->remove()` function is a configuration argument similar to that previously taken by `db_appinit` and now taken by the `DB_ENV->open()` method. For backward compatibility this new argument should simply be set to `NULL`. The force argument to `XXX_unlink` is now a flag value that is set by bitwise inclusively **OR**'ing it the `DB_ENV->remove()` flag argument.

function arguments

In Berkeley DB 3.0, there are no longer separate structures that represent each subsystem (for example, DB_LOCKTAB or DB_TXNMGR), and an overall DB_ENV environment structure. Instead there is only the DB_ENV references should be passed around by your application instead of passing around DB_LOCKTAB or DB_TXNMGR references.

Each of the following functions:

```
lock_detect
lock_get
lock_id
lock_put
lock_stat
lock_vec
```

should have its first argument, a reference to the DB_LOCKTAB structure, replaced with a reference to the enclosing DB_ENV structure. For example, the following line of code from a Berkeley DB 2.X application:

```
DB_LOCKTAB *lt;
DB_LOCK lock;

ret = lock_put(lt, lock);
```

should now be written as follows:

```
DB_ENV *dbenv;
DB_LOCK *lock;

ret = lock_put(dbenv, lock);
```

Similarly, all of the functions:

```
log_archive
log_compare
log_file
log_flush
log_get
log_put
log_register
log_stat
log_unregister
```

should have their DB_LOG argument replaced with a reference to a DB_ENV structure, and the functions:

```
memp_fopen
memp_register
memp_stat
memp_sync
memp_trickle
```

should have their DB_MPOOL argument replaced with a reference to a DB_ENV structure.

You should remove all references to DB_LOCKTAB, DB_LOG, DB_MPOOL, and DB_TXNMGR structures from your application, they are no longer useful in any way. In fact, a simple way to identify all of the places that need to be upgraded is to remove all such structures and variables they declare, and then compile. You will see a warning message from your compiler in each case that needs to be upgraded.

DB_ENV structure

The DB_ENV structure is now opaque for applications in the Berkeley DB 3.0 release. Accesses to any fields within that structure by the application should be replaced with method calls. The following example illustrates this using the historic errpfx structure field. In the Berkeley DB 2.X releases, applications set error prefixes using code similar to the following:

```
DB_ENV *dbenv;

dbenv->errpfx = "my prefix";
```

in the Berkeley DB 3.X releases, this should be done using the DB_ENV->set_errpfx() method, as follows:

```
DB_ENV *dbenv;

dbenv->set_errpfx(dbenv, "my prefix");
```

The following table lists the DB_ENV fields previously used by applications and the methods that should now be used to set them.

DB_ENV field	Berkeley DB 3.X method
db_errcall	DB_ENV->set_errcall()
db_errfile	DB_ENV->set_errfile()
db_errpfx	DB_ENV->set_errpfx()
db_lorder	This field was removed from the DB_ENV structure in the Berkeley DB 3.0 release as no application should have ever used it. Any code using it should be evaluated for potential bugs.
db_paniccall	DB_ENV->set_paniccall
db_verbose	DB_ENV->set_verbose() Note: the db_verbose field was a simple boolean toggle, the DB_ENV->set_verbose() method takes arguments that specify exactly which verbose messages are desired.
lg_max	DB_ENV->set_lg_max()
lk_conflicts	DB_ENV->set_lk_conflicts()

DB_ENV field	Berkeley DB 3.X method
lk_detect	DB_ENV->set_lk_detect()
lk_max	dbenv->set_lk_max
lk_modes	DB_ENV->set_lk_conflicts()
mp_mmapsize	DB_ENV->set_mp_mmapsize()
mp_size	DB_ENV->set_cachesize() Note: the DB_ENV->set_cachesize() function takes additional arguments. Setting both the second argument (the number of GB in the pool) and the last argument (the number of memory pools to create) to 0 will result in behavior that is backward-compatible with previous Berkeley DB releases.
tx_info	This field was used by applications as an argument to the transaction subsystem functions. As those functions take references to a DB_ENV structure as arguments in the Berkeley DB 3.0 release, it should no longer be used by any application.
tx_max	DB_ENV->set_tx_max()
tx_recover	dbenv->set_tx_recover

database open/close

Database opens were changed in the Berkeley DB 3.0 release in a similar way to environment opens.

To upgrade your application, first find each place your application opens a database, that is, calls the db_open function. Each of these calls should be replaced with calls to db_create() and DB->open().

Here's an example creating a Berkeley DB database using the 2.X interface:

```
DB *dbp;
DB_ENV *dbenv;
int ret;

if ((ret = db_open(DATABASE,
    DB_BTREE, DB_CREATE, 0664, dbenv, NULL, &dbp)) != 0)
    return (ret);
```

In the Berkeley DB 3.0 release, this code would be written as:

```
DB *dbp;
DB_ENV *dbenv;
int ret;
```

```
if ((ret = db_create(&dbp, dbenv, 0)) != 0)
    return (ret);

if ((ret = dbp->open(dbp,
    DATABASE, NULL, DB_BTREE, DB_CREATE, 0664)) != 0) {
    (void)dbp->close(dbp, 0);
    return (ret);
}
```

As you can see, the arguments to `db_open` and to `DB->open()` are largely the same. There is some re-organization, and note that the enclosing `DB_ENV` structure is specified when the DB object is created using the `db_create()` function. There is one additional argument to `DB->open()`, argument #3. For backward compatibility with the 2.X Berkeley DB releases, simply set that argument to `NULL`.

There are two additional issues with the `db_open` call.

First, it was possible in the 2.X releases for an application to provide an environment that did not contain a shared memory buffer pool as the database environment, and Berkeley DB would create a private one automatically. This functionality is no longer available, applications must specify the `DB_INIT_MPOOL` flag if databases are going to be opened in the environment.

The final issue with upgrading the `db_open` call is that the `DB_INFO` structure is no longer used, having been replaced by individual methods on the DB handle. That change is discussed in detail later in this chapter.

db_xa_open

The following change applies only to applications using Berkeley DB as an XA Resource Manager. If your application is not using Berkeley DB in this way, you can ignore this change.

The `db_xa_open` function has been replaced with the `DB_XA_CREATE` flag to the `db_create()` function. All calls to `db_xa_open` should be replaced with calls to `db_create()` with the `DB_XA_CREATE` flag set, followed by a call to the `DB->open()` function.

A similar change has been made for the C++ API, where the `DB_XA_CREATE` flag should be specified to the `Db` constructor. All calls to the `Db::xa_open` method should be replaced with the `DB_XA_CREATE` flag to the `Db` constructor, followed by a call to the `DB::open` method.

DB structure

The DB structure is now opaque for applications in the Berkeley DB 3.0 release. Accesses to any fields within that structure by the application should be replaced with method calls. The following example illustrates this using the historic type structure field. In the Berkeley DB 2.X releases, applications could find the type of an underlying database using code similar to the following:

```
DB *db;
DB_TYPE type;
```

```
type = db->type;
```

in the Berkeley DB 3.X releases, this should be done using the `DB->get_type()` method, as follows:

```
DB *db;
DB_TYPE type;

type = db->get_type(db);
```

The following table lists the DB fields previously used by applications and the methods that should now be used to get or set them.

DB field	Berkeley DB 3.X method
byteswapped	<code>DB->get_byteswapped()</code>
db_errcall	<code>DB->set_errcall()</code>
db_errfile	<code>DB->set_errfile()</code>
db_errpfx	<code>DB->set_errpfx()</code>
db_paniccall	<code>DB->set_paniccall</code>
type	<code>DB->get_type()</code>

DBINFO structure

The `DB_INFO` structure has been removed from the Berkeley DB 3.0 release. Accesses to any fields within that structure by the application should be replaced with method calls on the DB handle. The following example illustrates this using the historic `db_cachesize` structure field. In the Berkeley DB 2.X releases, applications could set the size of an underlying database cache using code similar to the following:

```
DB_INFO dbinfo;

memset(&dbinfo, 0, sizeof(dbinfo));
dbinfo.db_cachesize = 1024 * 1024;
```

in the Berkeley DB 3.X releases, this should be done using the `DB->set_cachesize()` method, as follows:

```
DB *db;
int ret;

ret = db->set_cachesize(db, 0, 1024 * 1024, 0);
```

The `DB_INFO` structure is no longer used in any way by the Berkeley DB 3.0 release, and should be removed from the application.

The following table lists the `DB_INFO` fields previously used by applications and the methods that should now be used to set them. Because these calls provide configuration for the database open, they must precede the call to `DB->open()`. Calling them after the call to `DB->open()` will return an error.

DB_INFO field	Berkeley DB 3.X method
bt_compare	DB->set_bt_compare()
bt_minkey	DB->set_bt_minkey()
bt_prefix	DB->set_bt_prefix()
db_cachesize	DB->set_cachesize() Note: the DB->set_cachesize() function takes additional arguments. Setting both the second argument (the number of GB in the pool) and the last argument (the number of memory pools to create) to 0 will result in behavior that is backward-compatible with previous Berkeley DB releases.
db_lorder	DB->set_lorder()
db_malloc	DB->set_malloc
db_pagesize	DB->set_pagesize()
dup_compare	DB->set_dup_compare()
flags	DB->set_flags() Note: the DB_DELIMITER, DB_FIXEDLEN and DB_PAD flags no longer need to be set as there are specific methods off the DB handle that set the file delimiter, the length of fixed-length records and the fixed-length record pad character. They should simply be discarded from the application.
h_ffactor	DB->set_h_ffactor()
h_hash	DB->set_h_hash()
h_nelem	DB->set_h_nelem()
re_delim	DB->set_re_delim()
re_len	DB->set_re_len()
re_pad	DB->set_re_pad()
re_source	DB->set_re_source()

DB->join

Historically, the last two arguments to the DB->join() method were a flags value followed by a reference to a memory location to store the returned cursor object. In the Berkeley DB 3.0 release, the order of those two arguments has been swapped for consistency with other Berkeley DB interfaces.

The application should be searched for any occurrences of DB->join(). For each of these, the order of the last two arguments should be swapped.

DB->stat

The `bt_flags` field returned from the `DB->stat()` method for Btree and Recno databases has been removed, and this information is no longer available.

DB->sync and DB->close

In previous Berkeley DB releases, the `DB->close()` and `DB->sync()` methods discarded any return of `DB_INCOMPLETE` from the underlying buffer pool interfaces, and returned success to its caller. (The `DB_INCOMPLETE` error will be returned if the buffer pool functions are unable to flush all of the database's dirty blocks from the pool. This often happens if another thread is reading or writing the database's pages in the pool.)

In the 3.X release, `DB->sync()` and `DB->close()` will return `DB_INCOMPLETE` to the application. The best solution is to not call `DB->sync()` with the `DB_NOSYNC` flag to the `DB->close()` method when multiple threads are expected to be accessing the database. Alternatively, the caller can ignore any error return of `DB_INCOMPLETE`.

lock_put

An argument change has been made in the `lock_put` function.

The application should be searched for any occurrences of `lock_put`. For each one, instead of passing a `DB_LOCK` variable as the last argument to the function, the address of the `DB_LOCK` variable should be passed.

lock_detect

An additional argument has been added to the `lock_detect` function.

The application should be searched for any occurrences of `lock_detect`. For each one, a `NULL` argument should be appended to the current arguments.

lock_stat

The `st_magic`, `st_version`, `st_numobjs` and `st_refcnt` fields returned from the `lock_stat` function have been removed, and this information is no longer available.

log_register

An argument has been removed from the `log_register` function. The application should be searched for any occurrences of `log_register`. In each of these, the `DBTYPE` argument (it is the fourth argument) should be removed.

log_stat

The `st_refcnt` field returned from the `log_stat` function has been removed, and this information is no longer available.

memp_stat

The **st_refcnt** field returned from the `memp_stat` function has been removed, and this information is no longer available.

The **st_cachesize** field returned from the `memp_stat` function has been replaced with two new fields, **st_gbytes** and **st_bytes**.

txn_begin

An additional argument has been added to the `txn_begin` function.

The application should be searched for any occurrences of `txn_begin`. For each one, an argument of 0 should be appended to the current arguments.

txn_commit

An additional argument has been added to the `txn_commit` function.

The application should be searched for any occurrences of `txn_commit`. For each one, an argument of 0 should be appended to the current arguments.

txn_stat

The **st_refcnt** field returned from the `txn_stat` function has been removed, and this information is no longer available.

DB_RMW

The following change applies only to applications using the Berkeley DB Concurrent Data Store product. If your application is not using that product, you can ignore this change.

Historically, the `DB->cursor()` method took the `DB_RMW` flag to indicate that the created cursor would be used for write operations on the database. This flag has been renamed to the `DB_WRITECURSOR` flag.

The application should be searched for any occurrences of `DB_RMW`. For each of these, any that are arguments to the `DB->cursor()` function should be changed to pass in the `DB_WRITECURSOR` flag instead.

DB_LOCK_NOTHELD

Historically, the Berkeley DB `lock_put` and `lock_vec` interfaces could return the `DB_LOCK_NOTHELD` error to indicate that a lock could not be released as it was held by another locker. This error can no longer be returned under any circumstances. The application should be searched for any occurrences of `DB_LOCK_NOTHELD`. For each of these, the test and any error processing should be removed.

EAGAIN

Historically, the Berkeley DB interfaces have returned the POSIX error value EAGAIN to indicate a deadlock. This has been removed from the Berkeley DB 3.0 release in order to make it possible for applications to distinguish between EAGAIN errors returned by the system and returns from Berkeley DB indicating deadlock.

The application should be searched for any occurrences of EAGAIN. For each of these, any that are checking for a deadlock return from Berkeley DB should be changed to check for the DB_LOCK_DEADLOCK return value.

If, for any reason, this is a difficult change for the application to make, the `include/db.src` distribution file should be modified to translate all returns of DB_LOCK_DEADLOCK to EAGAIN. Search for the string EAGAIN in that file, there is a comment that describes how to make the change.

EACCES

There was an error in previous releases of the Berkeley DB documentation that said that the `lock_put` and `lock_vec` interfaces could return EACCES as an error to indicate that a lock could not be released because it was held by another locker. The application should be searched for any occurrences of EACCES. For each of these, any that are checking for an error return from `lock_put` or `lock_vec` should have the test and any error handling removed.

db_jump_set

The `db_jump_set` interface has been removed from the Berkeley DB 3.0 release, replaced by method calls on the DB_ENV handle.

The following table lists the `db_jump_set` arguments previously used by applications and the methods that should now be used instead.

db_jump_set argument	Berkeley DB 3.X method
DB_FUNC_CLOSE	db_env_set_func_close
DB_FUNC_DIRFREE	db_env_set_func_dirfree
DB_FUNC_DIRLIST	db_env_set_func_dirlist
DB_FUNC_EXISTS	db_env_set_func_exists
DB_FUNC_FREE	db_env_set_func_free
DB_FUNC_FSYNC	db_env_set_func_fsync
DB_FUNC_IOINFO	db_env_set_func_ioinfo
DB_FUNC_MALLOC	db_env_set_func_malloc
DB_FUNC_MAP	dbenv_set_func_map
DB_FUNC_OPEN	db_env_set_func_open
DB_FUNC_READ	db_env_set_func_read

db_jump_set argument	Berkeley DB 3.X method
DB_FUNC_REALLOC	db_env_set_func_realloc
DB_FUNC_RUNLINK	The DB_FUNC_RUNLINK functionality has been removed from the Berkeley DB 3.0 release, and should be removed from the application.
DB_FUNC_SEEK	db_env_set_func_seek
DB_FUNC_SLEEP	db_env_set_func_sleep
DB_FUNC_UNLINK	db_env_set_func_unlink
DB_FUNC_UNMAP	dbenv_set_func_unmap
DB_FUNC_WRITE	db_env_set_func_write
DB_FUNC_YIELD	db_env_set_func_yield

db_value_set

The db_value_set function has been removed from the Berkeley DB 3.0 release, replaced by method calls on the DB_ENV handle.

The following table lists the db_value_set arguments previously used by applications and the function that should now be used instead.

db_value_set argument	Berkeley DB 3.X method
DB_MUTEX_LOCKS	dbenv_set_mutexlocks
DB_REGION_ANON	The DB_REGION_ANON functionality has been replaced by the DB_SYSTEM_MEM and DB_PRIVATE flags to the DB_ENV->open() function. A direct translation is not available, please review the DB_ENV->open() manual page for more information.
DB_REGION_INIT	dbenv_set_region_init
DB_REGION_NAME	The DB_REGION_NAME functionality has been replaced by the DB_SYSTEM_MEM and DB_PRIVATE flags to the DB_ENV->open() function. A direct translation is not available, please review the DB_ENV->open() manual page for more information.
DB_TSL_SPINS	dbenv_set_tas_spins

DbEnv class for C++ and Java

The DbEnv::appinit() method and two constructors for the DbEnv class are gone. There is now a single way to create and initialize the environment. The way to create an environment is to use the new DbEnv constructor with one argument. After this call, the DbEnv can be configured with various set_XXX methods. Finally, a call to DbEnv::open is made to initialize the environment.

Here's a C++ example creating a Berkeley DB environment using the 2.X interface

```
int dberr;
DbEnv *dbenv = new DbEnv();

dbenv->set_error_stream(&cerr);
dbenv->set_errpfx("myprog");

if ((dberr = dbenv->appinit("/database/home",
    NULL, DB_CREATE | DB_INIT_LOCK | DB_INIT_MPOOL)) != 0) {
    cerr << "failure: " << strerror(dberr);
    exit (1);
}
```

In the Berkeley DB 3.0 release, this code would be written as:

```
int dberr;
DbEnv *dbenv = new DbEnv(0);

dbenv->set_error_stream(&cerr);
dbenv->set_errpfx("myprog");

if ((dberr = dbenv->open("/database/home",
    NULL, DB_CREATE | DB_INIT_LOCK | DB_INIT_MPOOL, 0)) != 0) {
    cerr << "failure: " << dbenv->strerror(dberr);
    exit (1);
}
```

Here's a Java example creating a Berkeley DB environment using the 2.X interface:

```
int dberr;
DbEnv dbenv = new DbEnv();

dbenv.set_error_stream(System.err);
dbenv.set_errpfx("myprog");

dbenv.appinit("/database/home",
    null, Db.DB_CREATE | Db.DB_INIT_LOCK | Db.DB_INIT_MPOOL);
```

In the Berkeley DB 3.0 release, this code would be written as:

```
int dberr;
DbEnv dbenv = new DbEnv(0);

dbenv.set_error_stream(System.err);
dbenv.set_errpfx("myprog");

dbenv.open("/database/home",
    null, Db.DB_CREATE | Db.DB_INIT_LOCK | Db.DB_INIT_MPOOL, 0);
```

In the Berkeley DB 2.X release, DbEnv had accessors to obtain "managers" of type DbTxnMgr, DbMpool, DbLog, DbTxnMgr. If you used any of these managers, all their methods are now found directly in the DbEnv class.

Db class for C++ and Java

The static `Db::open` method and the `DbInfo` class have been removed in the Berkeley DB 3.0 release. The way to open a database file is to use the new `Db` constructor with two arguments, followed by `set_XXX` methods to configure the `Db` object, and finally a call to the new (nonstatic) `Db::open()`. In comparing the Berkeley DB 3.0 release open method with the 2.X static open method, the second argument is new. It is a database name, which can be null. The `DbEnv` argument has been removed, as the environment is now specified in the constructor. The open method no longer returns a `Db`, since it operates on one.

Here's a C++ example opening a Berkeley DB database using the 2.X interface:

```
// Note: by default, errors are thrown as exceptions
Db *table;
Db::open("lookup.db", DB_BTREE, DB_CREATE, 0644, dbenv, 0, &table);
```

In the Berkeley DB 3.0 release, this code would be written as:

```
// Note: by default, errors are thrown as exceptions
Db *table = new Db(dbenv, 0);
table->open("lookup.db", NULL, DB_BTREE, DB_CREATE, 0644);
```

Here's a Java example opening a Berkeley DB database using the 2.X interface:

```
// Note: errors are thrown as exceptions
Db table = Db.open("lookup.db", Db.DB_BTREE, Db.DB_CREATE, 0644, dbenv, 0);
```

In the Berkeley DB 3.0 release, this code would be written as:

```
// Note: errors are thrown as exceptions
Db table = new Db(dbenv, 0);
table.open("lookup.db", null, Db.DB_BTREE, Db.DB_CREATE, 0644);
```

Note that if the `dbenv` argument is null, the database will not exist within an environment.

additional C++ changes

The `Db::set_error_model` method is gone. The way to change the C++ API to return errors rather than throw exceptions is via a flag on the `DbEnv` or `Db` constructor. For example:

```
int dberr;
DbEnv *dbenv = new DbEnv(DB_CXX_NO_EXCEPTIONS);
```

creates an environment that will never throw exceptions, and method returns should be checked instead.

There are a number of smaller changes to the API that bring the C, C++ and Java APIs much closer in terms of functionality and usage. Please refer to the pages for upgrading C applications for further details.

additional Java changes

There are several additional types of exceptions thrown in the Berkeley DB 3.0 Java API.

DbMemoryException and DbDeadlockException can be caught independently of DbException if you want to do special handling for these kinds of errors. Since they are subclassed from DbException, a try block that catches DbException will catch these also, so code is not required to change. The catch clause for these new exceptions should appear before the catch clause for DbException.

You will need to add a catch clause for java.io.FileNotFoundException, since that can be thrown by Db.open and DbEnv.open.

There are a number of smaller changes to the API that bring the C, C++ and Java APIs much closer in terms of functionality and usage. Please refer to the pages for upgrading C applications for further details.

Upgrade Requirements

Log file formats and the Btree, Recno and Hash Access Method database formats changed in the Berkeley DB 3.0 release. (The on-disk Btree/Recno format changed from version 6 to version 7. The on-disk Hash format changed from version 5 to version 6.) Until the underlying databases are upgraded, the DB->open() method will return a DB_OLD_VERSION error.

For further information on upgrading Berkeley DB installations, see [Upgrading from previous versions of Berkeley DB \(page 2\)](#).

Chapter 15. Upgrading Berkeley DB 1.85 or 1.86 applications to Berkeley DB 2.0

Introduction

The following pages describe how to upgrade applications coded against the Berkeley DB 1.85 and 1.86 release interfaces to the Berkeley DB 2.0 release interfaces. They do not describe how to upgrade to the current Berkeley DB release interfaces.

It is not difficult to upgrade Berkeley DB 1.85 applications to use the Berkeley DB version 2 library. The Berkeley DB version 2 library has a Berkeley DB 1.85 compatibility API, which you can use by either recompiling your application's source code or by relinking its object files against the version 2 library. The underlying databases must be converted, however, as the Berkeley DB version 2 library has a different underlying database format.

System Integration

1. It is possible to maintain both the Berkeley DB 1.85 and Berkeley DB version 2 libraries on your system. However, the `db.h` include file that was distributed with Berkeley DB 1.85 is not compatible with the `db.h` file distributed with Berkeley DB version 2, so you will have to install them in different locations. In addition, both the Berkeley DB 1.85 and Berkeley DB version 2 libraries are named `libdb.a`.

As the Berkeley DB 1.85 library did not have an installation target in the Makefile, there's no way to know exactly where it was installed on the system. In addition, many vendors included it in the C library instead of as a separate library, and so it may actually be part of `libc` and the `db.h` include file may be installed in `/usr/include`.

For these reasons, the simplest way to maintain both libraries is to install Berkeley DB version 2 in a completely separate area of your system. The Berkeley DB version 2 installation process allows you to install into a standalone directory hierarchy on your system. See the Berkeley DB Installation and Build Guide for more information and instructions on how to install the Berkeley DB version 2 library, include files and documentation into specific locations.

2. Alternatively, you can replace Berkeley DB 1.85 on your system with Berkeley DB version 2. In this case, you'll probably want to install Berkeley DB version 2 in the normal place on your system, wherever that may be, and delete the Berkeley DB 1.85 include files, manual pages and libraries.
To replace 1.85 with version 2, you must either convert your 1.85 applications to use the version 2 API or build the Berkeley DB version 2 library to include Berkeley DB 1.85 interface compatibility code. Whether converting your applications to use the version 2 interface or using the version 1.85 compatibility API, you will need to recompile or relink your 1.85 applications, and you must convert any persistent application databases to the Berkeley DB version 2 database formats.

If you want to recompile your Berkeley DB 1.85 applications, you will have to change them to include the file `db_185.h` instead of `db.h`. (The `db_185.h` file is automatically

installed during the Berkeley DB version 2 installation process.) You can then recompile the applications, linking them against the Berkeley DB version 2 library.

For more information on compiling the Berkeley DB 1.85 compatibility code into the Berkeley DB version 2 library, see Berkeley DB Installation and Build Guide.

For more information on converting databases from the Berkeley DB 1.85 formats to the Berkeley DB version 2 formats, see the `db_dump185` utility and the `db_load` utility documentation.

3. Finally, although we certainly do not recommend it, it is possible to load both Berkeley DB 1.85 and Berkeley DB version 2 into the same library. Similarly, it is possible to use both Berkeley DB 1.85 and Berkeley DB version 2 within a single application, although it is not possible to use them from within the same file.

The name space in Berkeley DB version 2 has been changed from that of previous Berkeley DB versions, notably version 1.85, for portability and consistency reasons. The only name collisions in the two libraries are the names used by the historic `dbm` and `hsearch` interfaces, and the Berkeley DB 1.85 compatibility interfaces in the Berkeley DB version 2 library.

If you are loading both Berkeley DB 1.85 and Berkeley DB version 2 into a single library, remove the historic interfaces from one of the two library builds, and configure the Berkeley DB version 2 build to not include the Berkeley DB 1.85 compatibility API, otherwise you could have collisions and undefined behavior. This can be done by editing the library Makefiles and reconfiguring and rebuilding the Berkeley DB version 2 library. Obviously, if you use the historic interfaces, you will get the version in the library from which you did not remove them. Similarly, you will not be able to access Berkeley DB version 2 files using the Berkeley DB 1.85 compatibility interface, since you have removed that from the library as well.

Converting Applications

Mapping the Berkeley DB 1.85 functionality into Berkeley DB version 2 is almost always simple. The manual page `DB->open()` replaces the Berkeley DB 1.85 manual pages `dbopen(3)`, `btree(3)`, `hash(3)` and `recno(3)`. You should be able to convert each 1.85 function call into a Berkeley DB version 2 function call using just the `DB->open()` documentation.

Some guidelines and things to watch out for:

1. Most access method functions have exactly the same semantics as in Berkeley DB 1.85, although the arguments to the functions have changed in some cases. To get your code to compile, the most common change is to add the transaction ID as an argument (`NULL`, since Berkeley DB 1.85 did not support transactions.)
2. You must always initialize DBT structures to zero before using them with any Berkeley DB version 2 function. (They do not normally have to be reinitialized each time, only when they are first allocated. Do this by declaring the DBT structure external or static, or by calling the C library routine `bzero(3)` or `memset(3)`.)
3. The error returns are completely different in the two versions. In Berkeley DB 1.85, `< 0` meant an error, and `> 0` meant a minor Berkeley DB exception. In Berkeley DB 2.0, `> 0`

means an error (the Berkeley DB version 2 functions return `errno` on error) and `< 0` means a Berkeley DB exception. See Program returns to applications for more information.

4. The Berkeley DB 1.85 `DB->seq` function has been replaced by cursors in Berkeley DB version 2. The semantics are approximately the same, but cursors require the creation of an extra object (the DBC object), which is then used to access the database. Specifically, the partial key match and range search functionality of the `R_CURSOR` flag in `DB->seq` has been replaced by the `DB_SET_RANGE` flag in `DBC->get()`.
5. In version 2 of the Berkeley DB library, additions or deletions into Recno (fixed and variable-length record) databases no longer automatically logically renumber all records after the add/delete point, by default. The default behavior is that deleting records does not cause subsequent records to be renumbered, and it is an error to attempt to add new records between records already in the database. Applications wanting the historic Recno access method semantics should call the `DB->set_flags()` method with the `DB_RENUMBER` flag.
6. Opening a database in Berkeley DB version 2 is a much heavier-weight operation than it was in Berkeley DB 1.85. Therefore, if your historic applications were written to open a database, perform a single operation, and close the database, you may observe performance degradation. In most cases, this is due to the expense of creating the environment upon each open. While we encourage restructuring your application to avoid repeated opens and closes, you can probably recover most of the lost performance by simply using a persistent environment across invocations.

While simply converting Berkeley DB 1.85 function calls to Berkeley DB version 2 function calls will work, we recommend that you eventually reconsider your application's interface to the Berkeley DB database library in light of the additional functionality supplied by Berkeley DB version 2, as it is likely to result in enhanced application performance.

Upgrade Requirements

You will need to upgrade your on-disk databases, as all access method database formats changed in the Berkeley DB 2.0 release. For information on converting databases from Berkeley DB 1.85 to Berkeley DB 2.0, see the `db_dump185` utility and `db_load` utility documentation. As database environments did not exist prior to the 2.0 release, there is no question of upgrading existing database environments.