OpenClovis

OpenClovis
Software Development Kit (SDK)
Service Description and API Reference for Database Abstraction Layer (DBAL) Management Service

For OpenClovis SDK Release 2.3 V0.4
Document Revision Date: March 08, 2007

Note: This document is not subject of the GPL license, even if you have obtained this document as a part of the GPL-ed version of OpenClovis SDK.

Contents

Chapter 1

Functional Overview

The OpenClovis Database Abstraction Layer (DBAL) provides a standard interface for any OpenClovis ASP infrastructure component or application to interface with the commonly used relational database.

OpenClovis ASP R2.3 currently supports the following databases:

- GNU Database Manager
- Berkeley Database
- SOLID Database

DBAL Library can be used by any software component that requires huge data storage. As an abstraction layer, it interacts with different databases.

The primary user of this interface is the COR component that can dump or read its object repository to and from a database, for either persistent storage or offline processing. DBAL is also a standalone library with no dependencies on any other OpenClovis ASP components.

Chapter 2

Service Model

TBD

Chapter 3

Service APIs

3.1 Type Definitions

3.1.1 ClDBFileT

```
typedef char* ClDBFileT;
```

The type of an identifier for the file name.

3.1.2 ClDBNameT

```
typedef char* ClDBNameT;
```

The type of an indentifier for the name of the database.

3.1.3 cDBType_e

```
typedef enum cDBType_e{
        CL_DB_TYPE_HASH=0,
        CL_DB_TYPE_BTREE,
        CL_DB_MAX_TYPE
};
```

The `cDBType_e` enumeration contains the definition for the type of the database.

- *CL_DB_TYPE_HASH* - Hash table.
- *CL_DB_TYPE_BTREE* - B-tree.

3.1.4 cDBFlag_e

```
typedef enum {
        CL_DB_CREAT=0,
        CL_DB_OPEN,
```

CL_DB_APPEND,
CL_DB_MAX_FLAG
}cDBFlag_e;;

The values of the `cDBFlag_e` enumeration type contains the definition of the database flag.

3.1.5 ClDBFlagT

typedef enum cDBFlag_e ClDBFlagT;

The type of the flag specifying the opening type.

3.1.6 ClDBHandleT

typedef ClHandleT ClDBHandleT;

The type of the handle for the the database.

3.1.7 ClDBRecordT

typedef ClUint8T ClDBRecordT;*

The type of the handle for a record.

3.1.8 ClDBKeyT

typedef ClUint8T ClDBKeyT;*

The type of the handle for the key.

3.2 Library Life Cycle APIs

3.2.1 clDbalLibInitialize

clDbalLibInitialize

Synopsis:

Contains DBAL configuration module entry-point.

Header File:

clDbalCfg.h

Syntax:

```
ClRcT clDbalInitialize(void);
```

Parameters:

None.

Return values:

CL_OK: The function executed successfully.

Description:

This function contains the Database Abstraction Layer configuration module entry-point. This will be used to set the configuration parameters of the DBAL.

Library File:

libClDbal.a

Related Function(s):

clDbalLibFinalize

3.2.2 clDbalLibFinalize

clDbalLibFinalize

Synopsis:
Contains DBAL configuration module exit-point.

Header File:
clDbalCfg.h

Syntax:

```
        ClRcT clDbalLibFinalize(void);
```

Parameters:
None.

Return values:
CL_OK: The function executed successfully.

Description:
This function contains the Database Abstraction Layer configuration module exit-point.

Library File:
libClDbal.a

Related Function(s):
clDbalLibInitialize

3.3 Functional APIs

3.3.1 clDbalOpen

clDbalOpen

Synopsis:
Opens a database.

Header File:
clDbalApi.h

Syntax:

```
ClRcT  clDbalOpen(
                     CL_IN ClDBFileT dbFile,
             CL_IN ClDBNameT dbName,
             CL_IN ClDBFlagT dbFlag,
                     CL_IN ClUint32T maxKeySize,
                     CL_IN ClUint32T maxRecordSize,
                     CL_OUT ClDBHandleT* pDBHandle);
```

Parameters:

dbFile: (in) This parameter specifies the file name and path, where the database is to be created. If this parameter is NULL and the database is Berkeley Database, then an in-memory database is created. This parameter is ignored, if the database ID is GNU DBM or SOLID.

dbName: (in) Name of the database to open.

dbFlag: (in) This flag can accept the following two values:

- If `CL_DB_CREAT` is specified and the database already exists, then the existing database is deleted and a new database is created. If the database does not exist, then a new one is created.
- If `CL_DB_OPEN` is specified and the existing database is opened.

maxKeySize: (in) Maximum size of the key (in bytes) to be stored in the database. This parameter is valid only for SOLID database.

maxRecordSize: (in) Maximum size of the record (in bytes) to be stored in the database. This parameter is valid only for SOLID database.

pDBHandle: (out)Pointer to the variable of type `ClDBHandleT` in which the newly created database handle is returned.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pDBHandle` contains a NULL pointer.

CL_ERR_INVALID_PARAMETER: An invalid parameter has been passed to the function. A parameter is not set correctly.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_DB_ERROR: Failure in the underlying database.

Description:
This function opens a database instance on the specified database. Handle returned by this function will be used for further DBAL operations.

Library File:

ClDbal

Related Function(s):

clDbalClose

3.3.2 clDbalClose

clDbalClose

Synopsis:

Closes a database.

Header File:

clDbalApi.h

Syntax:

```
    ClRcT  clDbalClose(
                     CL_IN ClDBHandleT dbHandle);
```

Parameters:

dbHandle: (in) Handle of the database being closed.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

Description:

This function closes a database instance using its handle. In case of Berkeley Database, if the database being closed is an in-memory database, then all the data is lost once the database is closed.

Library File:

ClDbal

Related Function(s):

clDbalOpen

3.3.3 clDbalRecordInsert

clDbalRecordInsert

Synopsis:
Adds a record into the database.

Header File:
clDbalApi.h

Syntax:

```
ClRcT clDbalRecordInsert(
                              CL_IN  ClDBHandleT  dbHandle,
                  CL_IN  ClDBKeyT     dbKey,
                  CL_IN  ClUint32T    keySize,
                CL_IN  ClDBRecordT  dbRec,
                CL_IN  ClUint32T    recSize);
```

Parameters:

dbHandle: (in) Handle to the database into which the record is being added.

dbKey: (in) Record's key handle. In case of GDBM, the key must be a string.

keySize: (in) Size of the key.

dbRec: (in) Record handle. In case of GDBM, the record must be a string.

recSize: (in) Size of the record.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

CL_ERR_DUPLICATE: If an already existing key is added.

CL_ERR_DB_ERROR: Failure in the underlying database.

Description:
This function adds a record to a database known by its handle. If the specified key already exists in the database, then an error is returned.

Library File:
ClDbal

Related Function(s):
clDbalOpen,clDbalClose, clDbalRecordGet, clDbalRecordReplace, clDbalRecordDelete

3.3.4 clDbalRecordReplace

clDbalRecordReplace

Synopsis:

Replaces a record in the database.

Header File:

clDbalApi.h

Syntax:

```
    ClRcT clDbalRecordReplace(
                                        CL_IN ClDBHandleT dbHandle,
                                        CL_IN ClDBKeyT   dbKey,
                                        CL_IN ClUint32T  keySize,
                                        CL_IN ClDBRecordT  dbRec,
                                        CL_IN ClUint32T  recSize);
```

Parameters:

dbHandle: (in) Handle to the database to which the record is being added.

dbKey: (in) Record's key handle.

keySize: (in) Size of the key.

dbRec: (in) Record handle.

recSize: (in) Size of the record.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

CL_ERR_DB_ERROR: Failure in the underlying database.

Description:

This function replaces a record in a database known by its handle. If the specified key already exists in the database, then the record associated with it is replaced with the current record. If the key does not exist, then the record is added.

Library File:

ClDbal

Related Function(s):

clDbalOpen,clDbalClose, clDbalRecordGet, clDbalRecordInsert, clDbalRecordDelete

3.3.5 clDbalRecordGet

clDbalRecordGet

Synopsis:
Retrieves a record from the database.

Header File:
clDbalApi.h

Syntax:

```
    ClRcT clDbalRecordGet(
                              CL_IN ClDBHandleT dbHandle,
                              CL_IN ClDBKeyT  dbKey,
                              CL_IN ClUint32T  keySize,
                              CL_OUT ClDBRecordT* pDBRec,
                              CL_OUT ClUint32T* pRecSize);
```

Parameters:

dbHandle: (in) Handle to the database from which the record is being retrieved.

dbKey: (in) Key handle, whose associated record is being retrieved.

keySize: (in) Size of the key.

pDBRec: (out) Pointer to the record handle, in which the record is being returned. Memory allocation is done by DBAL but the user must free this memory using `clDbalRecordFree()` function.

pRecSize: (out) Pointer to the variable in which size of the record is being returned.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pDBRec` or `pRecSize` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

CL_ERR_DB_ERROR: Failure in the underlying database.

CL_ERR_NOT_EXIST: The key does not exist.

Description:
This function retrieves a record from the database using the database handle and the record's key. If no record is found on the key of the specified record, an error is returned.

Library File:
ClDbal

Related Function(s):
clDbalOpen, clDbalClose, clDbalRecordReplace, clDbalRecordInsert, clDbalRecordDelete

3.3.6 clDbalRecordDelete

clDbalRecordDelete

Synopsis:
Deletes a record from the database.

Header File:
clDbalApi.h

Syntax:

```
  ClRcT clDbalRecordDelete(
CL_IN ClDBHandleT dbHandle,
CL_IN ClDBKeyT dbKey,
CL_IN ClUint32T keySize);
```

Parameters:

dbHandle: (in) Handle to the database from which the record is being deleted.

dbKey: (in) Key handle of the record.

keySize: (in) Size of the key.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

CL_ERR_DB_ERROR: Failure in the underlying database.

CL_ERR_NOT_EXIST: The key does not exist.

Description:
This function is used to delete a record from a database using its handle and record's key. If the record is not found, the function returns an error.

Library File:
ClDbal

Related Function(s):
clDbalOpen,clDbalClose, clDbalRecordReplace, clDbalRecordInsert, clDbalRecordGet

3.3.7 clDbalFirstRecordGet

clDbalFirstRecordGet

Synopsis:

Returns the first key and the associated record from a database.

Header File:

clDbalApi.h

Syntax:

```
    ClRcT clDbalFirstRecordGet(
                                    CL_IN ClDBHandleT dbHandle,
                                    CL_OUT ClDBKeyT*  pDBKey,
                                    CL_OUT ClUint32T* pKeySize,
                                    CL_OUT ClDBRecordT* pDBRec,
                                    CL_OUT ClUint32T* pRecSize);
```

Parameters:

dbHandle: (in) Handle to the database from where the record is being retrieved.

pDBKey: (out) Pointer to the key handle, in which the first record's key is returned. Memory allocation is done by DBAL but the user must free this memory using `clDbalKeyFree()` function.

pKeySize: (out) Pointer to the variable in which size of the key is being returned.

pDBRec: (out) Pointer to the record handle in which the first record is returned. Memory allocation is done by DBAL but the user must free this memory using `clDbalRecordFree()` function.

pRecSize: (out) Pointer to the variable in which size of the record is being returned.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pDBKey`, `pKeySize`, `pDBRec`, or `pRecSize` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

CL_ERR_DB_ERROR: Failure in the underlying database.

CL_ERR_NOT_EXIST: The first record does not exist.

Description:

This function is used to return the first key and the associated record from a database. If no records are found in the specified database, an error is returned.

Library File:

ClDbal

Related Function(s):

clDbalOpen, clDbalClose, clDbalRecordReplace, clDbalRecordInsert, clDbalRecordGet, clDbalRecordDelete, clDbalNextRecordGet

3.3.8 clDbalNextRecordGet

clDbalNextRecordGet

Synopsis:

Returns the next key and the associated record from a database.

Header File:

clDbalApi.h

Syntax:

```
   ClRcT clDbalNextRecordGet(
CL_IN ClDBHandleT dbHandle,
                                        CL_IN ClDBKeyT currentKey,
                                        CL_IN ClUint32T currentKeySize,
                             CL_OUT ClDBKeyT* pDBNextKey,
                             CL_OUT ClUint32T* pNextKeySize,
                             CL_OUT ClDBRecordT* pDBNextRec,
                                        CL_OUT ClUint32T* pNextRecSize);
```

Parameters:

dbHandle: (in) Handle to the database from which the next record is being retrieved.

currentKey: (in) Key handle of the current record.

currentKeySize: (in) Size of the current key.

pDBNextKey: (out) Pointer to the key handle, in which the next record's key is being returned. Memory allocation is done by DBAL but the user must free this memory using `clDbalKeyFree()` function.

pNextKeySize: (out) Pointer to the variable in which size of the key is being returned.

pDBNextRec: (out) Pointer to the record handle, in which handle of the next record is returned. Memory allocation is done by DBAL but the user must free this memory using `clDbalRecordFree()` function.

pNextRecSize: (out) Pointer to the variable in which size of the record is being returned.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pDBNextRec`, `pNextRecSize`, `pNextKeySize`, or `pDBNextKey` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: dbHandle contains an invalid handle.

CL_ERR_DB_ERROR: Failure in the underlying database.

CL_ERR_NOT_EXIST: The key does not exist.

Description:

This function is used to retrieve the next key and the associated record from a database using the current record's key. If the record is not available, an error is returned.

Library File:

ClDbal

Related Function(s):

clDbalOpen, clDbalClose, clDbalRecordReplace, clDbalRecordInsert, clDbalRecordGet, clDbalRecordDelete,

3.3.9 clDbalTransactionBegin

clDbalTransactionBegin

Synopsis:

Begins a transaction.

Header File:

clDbalApi.h

Syntax:

```
    ClRcT clDbalTransactionBegin(
                            CL_IN ClDBHandleT  dbHandle);
```

Parameters:

dbHandle: (in) Handle to the database in which the transaction is to be started.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

CL_ERR_NOT_SUPPORTED: The underlying database does not support transactions.

Description:

This function marks the beginning of a transaction. This function must be called before calling further DBAL transaction related functions. This function returns `transactionId`, which will be passed for further operations as part of the `dbHandle`.

Library File:

ClDbal

Note:

Transactions may only span threads, if they do so serially. That is, each transaction must be active in only a single thread of control at a given instant.

Related Function(s):

clDbalTransactionCommit, clDbalTransactionAbort

3.3.10 clDbalTransactionCommit

clDbalTransactionCommit

Synopsis:

Commits a transaction.

Header File:

clDbalApi.h

Syntax:

```
    ClRcT clDbalTransactionCommit(
                                CL_IN ClDBHandleT  dbHandle);
```

Parameters:

dbHandle: (in) Handle to the database in which the transaction is being commited.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: On passing a NULL pointer.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

CL_ERR_COMMIT_FAILED: Commit of underlying database failed.

CL_ERR_NOT_SUPPORTED: The underlying database does not support transactions.

Description:

This function commits the recently started transaction. All operations performed after calling `clDbalTransactionBegin`, will be committed in the database.

Library File:

ClDbal

Related Function(s):

clDbalTransactionBegin, clDbalTransactionAbort

3.3.11 clDbalTransactionAbort

clDbalTransactionAbort

Synopsis:
Aborts a transaction.

Header File:
clDbalApi.h

Syntax:

```
    ClRcT clDbalTransactionAbort(
                                    ClDBHandleT  dbHandle);
```

Parameters:
dbHandle: (in) Handle to the database in which the transaction is being aborted.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

CL_ERR_ABORT_FAILED: Failure in the abort of the underlying database.

CL_ERR_NOT_SUPPORTED: The underlying database does not support transactions.

Description:
This function is used to abort the recently started transaction. This removes the transaction ID specified in the handle.

Library File:
ClDbal

Related Function(s):
clDbalTransactionBegin, clDbalTransactionCommit

3.3.12 clDbalRecordFree

clDbalRecordFree

Synopsis:

Frees the database record.

Header File:

clDbalApi.h

Syntax:

```
    ClRcT clDbalRecordFree(
                            CL_IN ClDBHandleT dbHandle
                            CL_IN ClDBRecordT dbRec);
```

Parameters:

dbHandle: (in) Handle to the database.

dbRec: (in) Record to be freed.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

Description:

This function is used to free the record returned by `clDbalFirstRecordGet()` and `clDbalNextRecordGet()` APIs. The record is created by `clDbalRecordInsert()` and cannot be freed by `clHeapFree()`. So to clean the records, this function must be called.

Library File:

ClDbal

Related Function(s):

clDbalRecordGet, clDbalFirstRecordGet, clDbalNextRecordGet, clDbalKeyFree

3.3.13 clDbalKeyFree

clDbalKeyFree

Synopsis:
Frees the database key.

Header File:
clDbalApi.h

Syntax:

```
ClRcT clDbalKeyFree(
                        CL_IN ClDBHandleT dbHandle
                          CL_IN ClDBKeyT dbKey);
```

Parameters:

dbHandle: (in) Handle to the database.

dbKey: (in) Key to be freed.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `dbHandle` is an invalid handle.

Description:
This function is used to free the Key returned by `clDbalFirstRecordGet()` and `clDbalNextRecordGet`. The key is created by `clDbalRecordInsert()` cannot be freed by `clHeapFree()`. So, this function needs to be called to free the keys, .

Library File:
ClDbal

Related Function(s):
clDbalFirstRecordGet, clDbalNextRecordGet

Chapter 4

Service Management Information Model

TBD

Chapter 5

Service Notifications

TBD

Chapter 6

Debug CLIs

TBD

Index