OpenClovis
Software Development Kit (SDK)
Service Description and API Reference for Transaction Management Service

For OpenClovis SDK Release 2.3 V0.5

Document Revision Date: March 27, 2007

Contents

Chapter 1

Functional Overview

The OpenClovis Transaction Manager (TM) provides an infrastructure library for resource managers to use transaction semantics to manage distributed data. Any component which requires to be part of transactions can link with this library and act as a resource manager. It automatically tracks participants and provides ACID semantics to ensure that all participants are updated or will rollback to previous state assuring data integrity despite component failures.

The TM supports transactions that can re-start. For instance, if any transaction participant fails, data recovery mechanism helps to recover the application state using the transaction logs for enhanced availability.

Chapter 2

Service Model

TBD

Chapter 3

Service APIs

3.1 Type Definitions

3.1.1 ClTxnAgentCallbacksT

typedef struct {
ClTxnAgentTxnJobCommitCallbackT fpTxnAgentJobCommit;
ClTxnAgentTxnJobPrepareCallbackT fpTxnAgentJobPrepare;
ClTxnAgentTxnJobRollbackCallbackT fpTxnAgentJobRollback;
} ClTxnAgentCallbacksT;

The structure, `ClTxnAgentCallbacksT`, contains a list of callback functions provided by the application for transaction agent for their role in active transactions. Application developer needs to initialize this structure and use it while calling `clTxnAgentServiceRegister()`.

3.1.2 ClTxnAgentServiceHandleT

typedef ClHandleT ClTxnAgentServiceHandleT;

Handle to the service that registers with the transaction- agent.

3.1.3 ClTxnTransactionCompletionCallbackT

*typedef void(*ClTxnTransactionCompletionCallbackT)(ClTxnTransactionHandleT txnHandle, ClRcT retCode);*

The type of callback function for the completion of an asynchronously initiated transaction. This callback is specified at the time of initialization of the client. The first parameter is the handle returned during the creation of the transaction. The second parameter is the return code indicating the status of the transaction (success or failure).

3.1.4 ClTxnTransactionHandleT

typedef ClHandleT ClTxnTransactionHandleT;

The type of handle used for an active transaction. This handle is returned during creation of a new transaction. It must be passed as the first parameter in all subsequent operations pertaining to this transaction session.

3.1.5 ClTxnClientHandleT

typedef ClHandleT ClTxnClientHandleT;

The type of the handle for a client accessing transaction service. This handle is returned while the transaction client is initialized. It must also be used while finalizing the transaction client.

3.1.6 ClTxnConfigMaskT

typedef ClUint8T ClTxnConfigMaskT;

The mask for specifying transaction configuration. This type must be passed as an argument during the creation of a new transaction.

3.1.7 ClTxnJobDefnHandleT

typedef ClHandleT ClTxnJobDefnHandleT;

The type of the handle used for user-defined job definition. This type is passed only during creation of the job.

3.1.8 ClTxnJobHandleT

typedef ClHandleT ClTxnJobHandleT;

The type of the handle of transaction job. This type is returned when a new transaction job is added. This type must be passed as an argument during subsequent operations pertaining to the job.

3.1.9 ClTxnCompConfigMaskT

typedef ClUint8T ClTxnCompConfigMaskT;

The type of mask for specifying component configuration. This type must be passed as an argument when you define information about the components participating in the job.

3.2 Library Life Cycle APIs

3.2.1 clTxnClientInitialize

clTxnClientInitialize

Synopsis:
Initializes transaction client library and registers callbacks.

Header File:
clTxnAgentApi.h

Syntax:

```
ClRcT clTxnClientInitialize(
                            CL_IN ClVersionT *pVersion,
                            CL_IN ClTxnTransactionCompletionCallbackT pTxnCallback,
                            CL_OUT ClTxnClientHandleT *pTxnLibHandle);
```

Parameters:

pVersion: (in) As an input parameter, `pVersion` is a pointer to the required version of Transaction library. As an output parameter, the version actually supported by the Transaction library is delivered.

pTxnCallback: (in) This is an optional parameter, if no asynchronous calls are made. This callback function is used to notify the completion of transaction-job.

pTxnLibHandle: (out) This handle identifies the initialization of the transaction library. This must be passed as the first input argument in all further invocation of functions related to the transaction library.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_NULL_POINTER: `pVersion` and `pTxnLibHandle` contains a NULL pointer.

CL_ERR_VERSION_MISMATCH: The version of the client and server is incompatible.

Description:
This function is used to initialize the transaction client library. The application component integrated with transaction client library uses this function to initialize the library. This function initializes various callback functions and performs version verification for invoking component.

Library File:
ClTxnAgent

Note:
Returned error is a combination of component-ID and error-code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve the error code.

Related Function(s):
clTxnAgentFinalize

3.2.2 clTxnClientFinalize

clTxnClient_Finalize

Synopsis:
Finalizes the transaction client library.

Header File:
clTxnAgentApi.h

Syntax:

```
ClRcT clTxnClientFinalize(
                                CL_IN ClTxnClientHandleT txnLibHandle);
```

Parameters

txnLibHandle:
(in) Handle of the transaction client library to be finalized obtained from the `clTxnClientInitialize()` function.

Return values:
CL_OK: The API executed successfully.

Description:
This function is used to finalize the transaction client library. It frees all information related to the transaction. It must be called when a component is terminated. To avoid memory leaks, every initialize call must be eventually followed by a finalize call.

Library File:
ClTxnAgent

Note:
Returned error is a combination of component ID and error code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve the error code.

Related Function(s):
clTxnAgentInitialize, clTxnAgentServiceUnRegister

3.3 Functional APIs

3.3.1 clTxnAgentServiceRegister

clTxnAgentServiceRegister

Synopsis:
Registers a service hosted in this component.

Header File:
clTxnAgentApi.h

Syntax:

```
ClRcT clTxnAgentServiceRegister(
                            CL_IN ClInt32T serviceId,
                            CL_IN ClTxnAgentCallbacksT tCallbacks,
                            CL_OUT ClTxnAgentServiceHandleT *pServiceHandle);
```

Parameters:

serviceId: (in) ID of the service hosted by this application component.

tCallbacks: (in) Instance of callback functions for this service.

pServiceHandle: (out) Handle to registered service in Transaction Agent.

Return values:

CL_OK: The API executed successfully.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_DUPLICATE: An entry for this service already exists.

Note:
Returned error is a combination of component ID and error code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve error code.

Description:
This function is used to register services hosted by a component. A component may consist of multiple services. A transaction requires multiple services of components for its completion. The Transaction Agent provides services to register the required transactions, using this function.

Library File:
ClTxnAgent

Related Function(s):
clTxnAgentServiceUnRegister

3.3.2 clTxnAgentServiceUnRegister

clTxnAgentServiceUnRegister

Synopsis:
Cancels the registration of services provided by the component for its role in transaction.

Header File:
clTxnAgentApi.h

Syntax:

```
ClRcT clTxnAgentServiceUnRegister(
                        CL_IN ClTxnAgentServiceHandleT serviceHandle);
```

Parameters:
serviceHandle: (in) Handle to the service has to be de-registered obtained from `lTxnAgentServiceRegister()`.

Return values:
CL_OK: The API executed successfully.

CL_ERR_NOT_EXIST: If the corresponding entry does not exist.

Note:
Returned error is a combination of component ID and error code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve error code.

Description:
This function is used to de-register the services of component from the Transaction Management. Applications register their role in transactions using `clTxnAgentServiceRegister()` and provide application-specific callback functions.

Library File:
ClTxnAgent

Related Function(s):
clTxnAgentServiceRegister

3.3.3 clTxnTransactionCreate

clTxnTransactionCreate

Synopsis:
Creates a new transaction.

Header File:
clTxnApi.h

Syntax:

```
ClRcT clTxnTransactionCreate(
                              CL_IN ClTxnConfigMaskT txnConfig,
                              CL_OUT ClTxnTransactionHandleT *pTxnHandle);
```

Parameters:

txnConfig: (in) Configuration for executing transaction-jobs.

pTxnHandle: (out) Handle to the newly created transaction. In all further operations pertaining to this transaction, this handle must be passed as the input argument.

Return values:

CL_OK: The API executed successfully.

CL_ERR_NULL_POINTER: `pTxnHandle` contains a NULL pointer.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_INVALID_PARAMETER: A parameter is not set correctly.

Note:
Returned error is a combination of component ID and error code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve error code.

Description:
This client function is used to create a new transaction in the transaction server. The new transaction created gets a unique identification which is used for reference. The application creates a new transaction and registers with the server before sending the request to execute the operation.
The Transaction management provides different run-time configuration that are specified while defining it.

Library File:
ClTxnClient

Related Function(s):
clTxnTransactionCancel , clTxnTransactionStart , clTxnTransactionStartAsync

3.3.4 clTxnTransactionCancel

clTxnTransactionCancel

Synopsis:
Cancels the given transaction.

Header File:
clTxnApi.h

Syntax:

```
ClRcT clTxnTransactionCancel(
                              CL_IN ClTxnTransactionHandleT txnHandle);
```

Parameters:
txnHandle: (in) Handle of the transaction to be deleted obtained from `clTxnTransactionCreate()`.

Return values:
CL_OK: The API executed successfully.

CL_ERR_NULL_POINTER: On passing a NULL pointer.

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

CL_ERR_TIMEOUT: The communication with the server failed.

Note:
Returned error is a combination of component ID and error code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve error code.

Description:
This client function is used to cancel the transaction in the transaction server. A transaction can be canceled, if it is in `PRE-INIT` state. The `PRE-INIT` state is when the jobs are being defined and `COMMIT` command has not been issued by the server to any one of the components participating in the transaction.

Library File:
ClTxnClient

Related Function(s):
clTxnTransactionCreate , clTxnTransactionStart , clTxnTransactionStartAsync

3.3.5 clTxnTransactionStart

clTxnTransactionStart

Synopsis:
Starts a transaction identified by transaction ID (asynchronously).

Header File:
clTxnApi.h

Syntax:

```
ClRcT clTxnTransactionStart(
                              CL_IN ClTxnTransactionHandleT txnHandle);
```

Parameters:
txnHandle: (in) Handle of transaction to be started obtained from `clTxnTransactionCreate()`.

Return values:
CL_OK: The API executed successfully.

CL_TXN_ERR_NOT_INITIALIZED: The transaction is not initialized.

CL_TXN_ERR_VALIDATE_FAILED: The application-defined preparation for the transaction failed.

CL_TXN_ERR_COMMIT_FAILED: The transaction resulted in aborting the operation.

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

CL_ERR_TIMEOUT: The communication with the server failed.

Note:
Returned error is a combination of component ID and error code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve error code.

Description:
This function is used to start a transaction identified by transaction ID. The client creates a new transaction to execute the intended operation. After the task is defined for this transaction, this function triggers the co-ordination to complete the transaction. This is a blocking call.

Library File:
ClTxnClient

Related Function(s):
clTxnTransactionCreate , clTxnTransactionCancel , clTxnTransactionStartAsync

3.3.6 clTxnTransactionStartAsync

clTxnTransactionStartAsync

Synopsis:
Asynchronously starts a transaction identified by the transaction ID.

Header File:
clTxnApi.h

Syntax:

```
ClRcT clTxnTransactionStartAsync(
                              CL_IN ClTxnTransactionHandleT txnHandle);
```

Parameters:
txnHandle: (in) Handle of the transaction to be started asynchronously. This is created by `clTxnTransactionCreate()`.

Return values:
CL_OK: The API executed successfully.

CL_TXN_ERR_NOT_INITIALIZED: The transaction is not initialized.

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

CL_ERR_TIMEOUT: The communication with the server failed.

Note:
Returned error is a combination of component ID and error code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve error code.

Description:
This function is used to start a transaction asynchronously identified by a transaction ID. The client creates a new transaction to execute the intended operation. Once the task is defined for this transaction, this function triggers the background coordination to complete the transaction. This is non-blocking.

Library File:
ClTxnClient

Related Function(s):
clTxnTransactionCreate , clTxnTransactionCancel , clTxnTransactionStart

3.3.7 clTxnJobAdd

clTxnJobAdd

Synopsis:
Registers a new job as part of transaction. A transaction consists of at least one job description.

Header File:
clTxnApi.h

Syntax:

```
ClRcT clTxnJobAdd(
                                CL_IN ClTxnTransactionHandleT      txnHandle,
                                CL_IN ClTxnJobDefnHandleT  jobDefn,
                                CL_IN ClUint32T             jobDefnSize,
                                CL_IN ClInt32T              serviceType,
                                CL_OUT ClTxnJobHandleT      *pJobHandle);
```

Parameters:

txnHandle: (in) Transaction handle to which the job is to be added. This is created by `clTxnTransactionCreate()` function.

jobDefn: (in) Application-defined job description.

jobDefnSize: (in) Size of application defined job-definition.

serviceType: (in) Service within component involved in transaction (-1 for all services of component).

pJobHandle: (out) Identification of job for later references.

Return values:

CL_OK: The API executed successfully.

CL_ERR_NULL_POINTER: `pJobHandle` contains a NULL pointer.

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

Note:
Returned error is a combination of component ID and error code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve error code.

Description:
This function is used to register a new job as a part of a transaction. As part of modifying a data entity in distributed environment, application provides necessary details termed as job-description. Flexibility is given to application developer to define jobs and pass-on definition as a byte-stream. Hence details about jobs are transparent to the transaction management. Each job is identified using a unique job ID.

Library File:
ClTxnClient

Related Function(s):
clTxnJobRemove

3.3.8 clTxnJobRemove

clTxnJobRemove

Synopsis:
Removes a registered job from the transaction.

Header File:
clTxnApi.h

Syntax:

```
ClRcT clTxnJobRemove(
                              CL_IN ClTxnTransactionHandleT txnHandle,
                              CL_IN ClTxnJobHandleT jobHandle);
```

Parameters:

txnHandle: (in) Transaction handle from which the job is to be removed. This handle is created by the `clTxnTransactionCreate()` function.

jobHandle: (in) Handle of the job to be deleted. This handle is created by the `clTxnJobAdd()` function.

Return values:

CL_OK: The API executed successfully.

CL_ERR_NOT_EXIST: There is no transaction or job corresponding to the handle passed to the function.

Note:
Returned error is a combination of component ID and error code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve error code.

Description:
This function is used to remove an already registered job from transaction. As part of the defining a transaction, application defines a transaction and manages jobs by registering and removing them. Each job registered in transaction is uniquely identified by a job ID.

Library File:
ClTxnClient

Related Function(s):
clTxnJobAdd

3.3.9 clTxnComponentSet

clTxnComponentSet

Synopsis:
Sets a component to be involved in transaction-job.

Header File:
clTxnApi.h

Syntax:

```
ClRcT clTxnComponentSet(
                      CL_IN ClTxnTransactionHandleT  txnHandle,
                      CL_IN ClTxnJobHandleT  jobHandle,
                      CL_IN ClIocAddressT txnCompAddress,
                      CL_IN ClTxnCompConfigMaskT  configMask);
```

Parameters:

txnHandle :(in) Transaction handle of the job for which components are being added. This is created by the `clTxnTransactionCreate()` function.

jobHandle: (in) Transaction-job handle for which components is to be added. This is created by `clTxnJobAdd()`.

txnCompAddress: (in) IOC address of the component.

configMask: (in) Configuration information defining the role of this component in transaction.

Return values:

CL_OK: The API executed successfully.

CL_ERR_NOT_EXIST: If there is no job corresponding to the handles passed.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_DUPLICATE: A duplicate entry exists.

Note:
Returned error is a combination of component ID and error code. Use `CL_GET_ERROR_CODE(RET_CODE)` defined in `clCommonErrors.h` to retrieve error code.

Description:
This function is used to set a component to be involved in a transaction-job. The application interfacing transaction-management using these client functions needs to provide a list of components involved in transaction. This function can also control the order in which these components are to be visited during execution of the transaction, as an option.

Note:
Currently, the list of components are for the job. The application must explicitly specify, if the transaction contains multiple jobs for each job. Order in which a component participates in a given transaction is dependent on the capabilities of all components (for 1-PC Capable, 2-PC Capable) and other factors related to resource availability (For example, locks).

Library File:
ClTxnClient

Related Function(s):
None.

Chapter 4

Service Management Information Model

TBD

Chapter 5

Service Notifications

TBD

Chapter 6

Configuration

TBD

Chapter 7

Debug CLIs

TBD

Index