OpenClovis

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

For OpenClovis SDK Release 2.3 V0.5

Document Revision Date: March 08, 2007

Contents

Chapter 1

Functional Overview

A Circular list is a chain of nodes, without any termination. The last node is linked back to the first node. A node is a data structure which contains user data. The following are the operations supported by Circular List implementation:

- Adds a node at the beginning of the list.
- Adds a node at the end of the list.
- Adds a node before a specific node in the list.
- Adds a node after a specific node in the list.
- Returns the first node from the list.
- Returns the last node from the list.
- Returns the previous node from the list.
- Returns the next node from the list.
- Walks through the list, starting from a specific node.
- Deletes a node from the list.
- Returns the number of nodes from the list.
- Retrieves the data from a node from the list.
- Destroys the list.

Before performing any of the above mentioned operations, you must create a list. While creating a list, you need to specify the maximum size for the list. If the maximum size is specified as 0, then you can add any number of nodes. Otherwise, the number of nodes you can add is limited to the maximum size. At any instant, the list can have a maximum of `maxSize` number of nodes, specified when the list is created.

This file contains the definitions and API prototypes for Circular Linked List.

1.1 Interaction with other components

Circular List APIs depend on Heap for memory allocation and functions to free the allocated memory.

Chapter 2

Service Model

TBD

Chapter 3

Service APIs

3.1 Type Definitions

3.1.1 ClClistDropPolicyT

```
typedef enum {
            CL_NO_DROP,
            CL_DROP_FIRST,
            CL_DROP_MAX_TYPE
} ClClistDropPolicyT;
```

The values of the *ClClistDropPolicyT* enumeration contains/decides the action to be taken when the linked list is full. The values of this enumeration have the following interpretation:

- *CL_NO_DROP* - The first node must not be dropped even if the list is full.
- *CL_DROP_FIRST* - Drops the first node, if the list is full.
- *CL_DROP_MAX_TYPE* - Adds a new drop policy type before the first node.

3.1.2 ClClistDeleteCallbackT

```
typedef void(*ClClistDeleteCallbackT)(
ClClistDataT userData);
```

The type of the callback function that is invoked when a node is deleted from the container. It deletes the callback function pointer.

3.1.3 ClClistT

```
typedef ClHandleT ClClistT;
```

The type of the handle for the circular list.

3.1.4 ClClistDataT

typedef ClHandleT ClClistDataT;

The type of the handle for the user-data.

3.1.5 ClClistNodeT

typedef ClHandleT ClClistNodeT;

The type of the handle for the circular node.

3.1.6 ClClistWalkCallbackT

*typedef void(*ClClistWalkCallbackT)(*
ClClistDataT userData,
*void *userArg);*

The type of the callback function to be used when iterating/walking through the elements of a container.

3.2 Functional APIs

3.2.1 clClistCreate

clClistCreate

Synopsis:
Creates a Circular Linked list.

Header File:
clClistApi.h

Syntax:

```
ClRcT clClistCreate(
                    CL_IN  ClUint32T maxSize,
                    CL_IN  ClClistDropPolicyT dropPolicy,
                    CL_IN  ClClistDeleteCallbackT fpUserDeleteCallBack,
                    CL_IN   ClClistDeleteCallbackT fpUserDestroyCallBack,
                    CL_OUT  ClClistT* pListHead);
```

Parameters:

maxSize: (in) Maximum size of the list. It specifies the maximum number of nodes that can exist at any instant in the list. This must be an unsigned integer. You can add nodes to the list until this maximum limit is reached. If the value of this parameter is set to 0, there is no limit on the size of the list.

dropPolicy: (in) Drop policy for the list. If you try to add a node into a list that has reached it maximum limit, and if the drop policy is set to `CL_DROP_FIRST`, the first node is dropped and the specified node is added. If this parameter is set to `CL_NO_DROP`, an error is returned. This parameter is valid, only if the list is of fixed size.

fpUserDeleteCallBack: (in) Pointer to the delete callback function of the user. This function accepts a parameter of type `ClClistDataT`. After deleting the specified node, the user-data stored in that node is passed as an argument to the callback function.

fpUserDestroyCallBacK: (in) Pointer to the destroy callback function. This function accepts a parameter of type `ClClistDataT`. When destroying, the user-data stored in each node is passed as an argument to the callback function, one by one.

pListHead: (out) Pointer to the variable of type `ClClistT` in which the function returns a valid list handle on successful creation of the list.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pListHead` contains a NULL pointer.

CL_ERR_NO_MEMORY: Memory allocation failure.

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

Description:
This function creates and initializes the Circular Linked list.

Library File:
libClUtils

Related Function(s):
clClistDelete

3.2.2 clClistFirstNodeAdd

clClistFirstNodeAdd

Synopsis:
Adds a node at the beginning of the list.

Header File:
clClistApi.h

Syntax:
```
    ClRcT clClistFirstNodeAdd(
                              CL_IN  ClClistT listHead,
                              CL_IN  ClClistDataT userData);
```

Parameters:

listHead: (in) Handle of the list returned by the `clClistCreate` API.

userData: (in) User-data. You must allocate and de-allocate memory for this parameter.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_MAXSIZE_REACHED: The maximum size is reached.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

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

Description:
This function adds a node at the beginning of the Circular Linked list.

Library File:
libClUtils

Related Function(s):
clClistLastNodeAdd, clClistAfterNodeAdd, clClistBeforeNodeAdd

3.2.3 clClistLastNodeAdd

clClistLastNodeAdd

Synopsis:
Adds a node at the end of the list.

Header File:
clClistApi.h

Syntax:

```
ClRcT clClistLastNodeAdd(
                        CL_IN  ClClistT listHead,
                        CL_IN  ClClistDataT userData);
```

Parameters:

listHead: (in) Handle of the list returned by the create API.

userData: (in) User-data. You must allocate and de-allocate memory for this parameter.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_MAXSIZE_REACHED: The maximum size is reached.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

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

Description:
This function adds a node at the end of the Circular Linked list.

Library File:
libClUtils

Related Function(s):
clClistAfterNodeAdd, clClistBeforeNodeAdd

3.2.4 clClistAfterNodeAdd

clClistAfterNodeAdd

Synopsis:
Adds a node after a specified node in the list.

Header File:
clClistApi.h

Syntax:

```
    ClRcT clClistAfterNodeAdd(
                            CL_IN  ClClistT listHead,
                            CL_IN  ClClistNodeT currentNode,
                            CL_IN  ClClistDataT userData);
```

Parameters:

listHead: (in) Handle of the list returned by the create API.

currentNode: (in) Handle of the node after which the specified data must be added.

userData: (in) User-data. You must allocate and de-allocate memory for this parameter.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_MAXSIZE_REACHED: The maximum size is reached.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

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

Description:
This function adds a node after a specified node in the Circular Linked list.

Library File:
libClUtils

Related Function(s):
clClistLastNodeAdd, clClistBeforeNodeAdd

3.2.5 clClistBeforeNodeAdd

clClistBeforeNodeAdd

Synopsis:
Adds a node before a specified node in the list.

Header File:
clClistApi.h

Syntax:

```
ClRcT  clClistBeforeNodeAdd(
                        CL_IN  ClClistT listHead,
                        CL_IN  ClClistNodeT currentNode,
                        CL_IN  ClClistDataT userData);
```

Parameters:

listHead: (in) Handle of the list returned by the create API.

currentNode: (in) Handle of the node before which the specified data must be added.

userData: (in) User-data. You must perform memory allocation and de-allocation for this parameter.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_MAXSIZE_REACHED: The maximum size is reached.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

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

Description:
This function adds a node before a specified node in the Circular Linked list.

Library File:
libClUtils

Related Function(s):
clClistLastNodeAdd, clClistAfterNodeAdd

3.2.6 clClistNodeDelete

clClistNodeDelete

Synopsis:
Deletes a node from the list.

Header File:
clClistApi.h

Syntax:

```
ClRcT clClistNodeDelete(
CL_IN ClClistT listHead,
CL_IN ClClistNodeT node);
```

Parameters:
listHead: (in) Handle of the list returned by the create API.

node: (in) Handle of the node which is to be deleted.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

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

Description:
This function deletes the specified node from the Circular Linked list. The delete callback function, registered during the creation is called with the data in the node to be deleted.

Library File:
libClUtils

Related Function(s):
clClistFirstNodeAdd, clClistLastNodeAdd, clClistAfterNodeAdd, clClistBeforeNodeAdd

3.2.7 clClistFirstNodeGet

clClistFirstNodeGet

Synopsis:
Returns the first node from the list.

Header File:
clClistApi.h

Syntax:

```
    ClRcT clClistFirstNodeGet(
                              CL_IN  ClClistT listHead,
                              CL_OUT  ClClistNodeT* pFirstNode);
```

Parameters:

listHead: (in) Handle of the list returned by the create API.

pFirstNode: (out) Pointer to the variable of type `ClClistNodeT`.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pFirstNode` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

CL_ERR_NOT_EXIST: The list is empty.

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

Description:
This function returns the first node from the Circular Linked list.

Library File:
libClUtils

Related Function(s):
clClistLastNodeGet, clClistNextNodeGet, clClistPreviousNodeGet

3.2.8 clClistLastNodeGet

clClistLastNodeGet

Synopsis:
Returns the last node from the list.

Header File:
clClistApi.h

Syntax:

```
ClRcT clClistLastNodeGet(
                        CL_IN  ClClistT listHead,
                        CL_OUT  ClClistNodeT* pLastNode);
```

Parameters:

listHead: (in) Handle of the list returned by the create API.

pLastNode: (out) Pointer to the variable of type `ClClistNodeT`.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pLastNode` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

CL_ERR_NOT_EXIST: The list is empty.

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

Description:
This function is used to return the last node from the Circular Linked list.

Library File:
libClUtils

Related Function(s):
clClistNextNodeGet, clClistPreviousNodeGet

3.2.9 clClistNextNodeGet

clClistNextNodeGet

Synopsis:
Returns the next node from the list.

Header File:
clClistApi.h

Syntax:

```
    ClRcT clClistNextNodeGet(
                           CL_IN  ClClistT listHead,
                           CL_IN  ClClistNodeT currentNode,
                           CL_OUT  ClClistNodeT* pNextNode);
```

Parameters:

listHead: (in) Handle of the list returned by the create API.

currentNode: (in) Handle of the current node.

pNextNode: (out) Pointer to the variable of type `ClClistNodeT`.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pNextNode` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

CL_ERR_NOT_EXIST: The list is empty.

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

Description:
This function is used to return the next node of the specified node from the Circular Linked list.

Library File:
libClUtils

Related Function(s):
clClistLastNodeGet, clClistPreviousNodeGet

3.2.10 clClistPreviousNodeGet

clClistPreviousNodeGet

Synopsis:
Returns the previous node from the list.

Header File:
clClistApi.h

Syntax:

```
ClRcT clClistPreviousNodeGet(
                           CL_IN  ClClistT listHead,
                           CL_IN  ClClistNodeT currentNode,
                           CL_OUT  ClClistNodeT* pPreviousNode);
```

Parameters:

listHead: (in) Handle of the list returned by the create API.

currentNode: (in) Handle of the current node.

pPreviousNode: (out) Pointer to the variable of type `ClClistNodeT`.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pPreviousNode` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

CL_ERR_NOT_EXIST: The list is empty.

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

Description:
This function is used to return the previous node of the specified node from the Circular Linked list.

Library File:
libClUtils

Related Function(s):
clClistLastNodeGet, clClistNextNodeGet

3.2.11 clClistWalk

clClistWalk

Synopsis:
Walks through the list.

Header File:
clClistApi.h

Syntax:

```
    ClRcT clClistWalk(
                    CL_IN  ClClistT listHead,
                    CL_IN  ClClistWalkCallbackT fpUserWalkCallBack,
                    CL_IN  void* userArg);
```

Parameters:

listHead: (in) Handle of the list returned by the create API.

fpUserWalkCallBack: (in) Pointer to the user callback function. It can have two values:

- The first parameter must be of type `ClClistDataT`.
- The second parameter must be of type void $*$.
  The user-data stored in each node is passed one-by-one as the first argument to the callback function.

userArg: (in) User-specified argument. This variable is passed as the second argument to the callback function.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `userArg` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

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

Description:
This function performs a walk through on the Circular Linked list. The user-specified callback function is called with every node's data.

Library File:
libClUtils

Related Function(s):
None.

3.2.12 clClistDataGet

clClistDataGet

Synopsis:
Retrieves data from a node in the list.

Header File:
clClistApi.h

Syntax:

```
ClRcT clClistDataGet(
                   CL_IN  ClClistT listHead,
                   CL_IN  ClClistNodeT node,
                   CL_OUT  ClClistDataT* pUserData);
```

Parameters:

listHead: (in) Handle of the list returned by the create API.

node: (in) Handle of the node.

pUserData: (out) Pointer to the variable of type `ClClistDataT`.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pUserData` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

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

Description:
This function retrieves data from the specified node in the list.

Library File:
libClUtils

Related Function(s):
None.

3.2.13 clClistSizeGet

clClistSizeGet

Synopsis:
Returns the number of data elements (nodes) in the list.

Header File:
clClistApi.h

Syntax:
```
    ClRcT clClistSizeGet(
                   CL_IN  ClClistT listHead,
                   CL_OUT  ClUint32T* pSize);
```

Parameters:
listHead: (in) Handle of the list returned by the create API.

pSize: (out) Pointer to the variable of type *ClUint32T*.

Return values:
CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pSize` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

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

Description:
This function is used to return the number of data elements (nodes) in the list.

Library File:
libClUtils

Related Function(s):
None.

3.2.14 clClistDelete

clClistDelete

Synopsis:
Destroys the list.

Header File:
clClistApi.h

Syntax:

```
ClRcT clClistDelete(
                    CL_IN  ClClistT* pListHead);
```

Parameters:
pListHead: (in) Handle of the list returned by the create API.

Return values:
CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pListHead` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: An invalid handle has been passed to the function.

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

Description:
This function deletes all the nodes in the list. The destroy callback function, registered during creation, is called for every node in the list with the corresponding data.

Library File:
libClUtils

Related Function(s):
clClistCreate

Chapter 4

Service Management Information Model

TBD

Chapter 5

Service Notifications

TBD

Chapter 6

Debug CLIs

TBD

Index