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

For OpenClovis SDK Release 2.3 V0.4

Document Revision Date: March 08, 2007

DRAFT

Copyright © 2007 OpenClovis Inc.

All rights reserved
This document contains proprietary and confidential information of OpenClovis Inc., and may not be used, modified, copied, reproduced, disclosed or distributed in whole or in part except as authorized by OpenClovis Inc. This document is intended for informational use and planning purposes only. All planned features, specifications, and content are subject to change without notice.

Third-Party Trademarks
Sun, Sun Microsystems, and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark of The Open Group. Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries. CLEI is a trademark of Telcordia Technologies, Inc. Adobe, Acrobat, and Acrobat Reader are registered trademarks of Adobe Systems, Inc. All other trademarks, service marks, product names, or brand names mentioned in this document are the property of their respective owners.

Government Use
Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in FAR 12.212 (Commercial Computer Software-Restricted Rights) and DFAR 227.7202 (Rights in Technical Data and Computer Software), as applicable.

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 Queue Library provides implementation for an ordered list. It supports enqueuing, dequeuing, retrieval of a node, and walk through the queue.

FIFO, another name for a queue, is an acronym for the way it operates, First-In-First-Out. The standard interface to the FIFO queue consists of `ENQUEUE()` function for adding new elements, and a DEQUEUE() function for removing the oldest element. Enqueue adds an element at the rear (end) of the queue. Dequeue removes an element from the front (start) of the queue. The following operations are supported on queues:

- Enqueues an element into the queue.
- Dequeues an element from the queue.
- Walks through the queue.
- Returns the number of elements in the queue.
- Deletes the Queue.

  Before peforming any of the mentioned operations, you must create a queue. When creating a queue, you need to specify the maximum size for the queue. If the maximum size is specified as 0, then you can enqueue any number of elements. Otherwise, the number of elements you can enqueue is limited to the maximum size. That is, at any instant, the queue can have a maximum of `maxSize` number of elements, specified when the queue is created.

1.1 Interaction with other components

Queue APIs depend on Heap for memory allocation and functions to free the memory.

Chapter 2

Service Model

TBD

Chapter 3

Service APIs

3.1 Type Definitions

3.1.1 ClQueueDequeueCallbackT

*typedef void (*ClQueueDequeueCallbackT)(*
ClQueueDataT userData);

The type of the callback function to dequeue user-data.

3.1.2 ClQueueT

typedef ClHandleT ClQueueT;

The type of the handle for the queue.

3.1.3 ClQueueDataT

typedef ClHandleT ClQueueDataT;

The type of the handle for the user-data.

3.1.4 ClQueueWalkCallbackT

*typedef void (*ClQueueWalkCallbackT)(*
ClQueueDataT userData,
void userArg);*

The type of the callback function for walking through the queue.

3.2 Functional APIs

3.2.1 clQueueCreate

clQueueCreate

Synopsis:
Creates a queue.

Header File:
clQueueApi.h

Syntax:

```
ClRcT clQueueCreate(
                          CL_IN  ClUint32T maxSize,
                          CL_IN  ClQueueDequeueCallbackT fpUserDequeueCallBack,
                          CL_IN  ClQueueDequeueCallbackT fpUserDestroyCallBack,
                          CL_OUT  ClQueueT* pQueueHead);
```

Parameters:

maxSize: (in) Maximum size of the Queue. It specifies the maximum number of elements that can exist at any instant in the Queue. This must be an unsigned integer. You can enqueue elements into the queue until this maximum limit is reached. If you specify this parameter as 0, then there is no limit on the size of the Queue.

fpUserDequeueCallBack: (in) Pointer to the dequeue callback function. This function accepts a parameter of type `ClQueueDataT`. The user-data of the removed element from the queue is passed as an argument to the callback function.

fpUserDestroyCallBack: (in) Pointer to the destroy callback function. This function accepts a parameter of type `ClQueueDataT`. This function is called for every element in the queue.

pQueueHead: (out) Pointer to the variable of type `ClQueueT` in which the function returns a valid Queue handle.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_NULL_POINTER: `pQueueHead` contains a NULL pointer.

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

Description:
This function is used to create and initialize a queue.

Library File:
libClUtils

Related Function(s):
clQueueDelete

3.2.2 clQueueNodeInsert

clQueueNodeInsert

Synopsis:
Enqueues an element (user-data) into the Queue.

Header File:
clQueueApi.h

Syntax:

```
ClRcT clQueueNodeInsert(
                    CL_IN  ClQueueT queueHead,
                    CL_IN  ClQueueDataT userData);
```

Parameters:
queueHead: (in) Handle of the queue returned by `clQueueCreate()` function.

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

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 is used to enqueue an element (user-data) into the queue.

Library File:
libClUtils

Related Function(s):
clQueueNodeDelete

3.2.3 clQueueNodeDelete

clQueueNodeDelete

Synopsis:
Removes an element from the queue.

Header File:
clQueueApi.h

Syntax:

```
ClRcT clQueueNodeDelete(
                    CL_IN  ClQueueT queueHead,
                    CL_OUT  ClQueueDataT* userData);
```

Parameters:

queueHead: (in) Handle of the queue returned by `clQueueCreate()` function.

userData: (out) Handle of the user-data. The user-data of the removed node is returned.

Return values:

CL_OK: The function executed successfully.

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

CL_ERR_NULL_POINTER: `userData` contains a NULL pointer.

CL_ERR_NOT_EXIST: The queue 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 remove an element from the front of the queue. The user delete callback, registered during creation, is called with the removed element (user-data).

Library File:
libClUtils

Related Function(s):
clQueueNodeInsert

3.2.4 clQueueWalk

clQueueWalk

Synopsis:
Walks through the queue.

Header File:
clQueueApi.h

Syntax:

```
ClRcT clQueueWalk(
                      CL_IN  ClQueueT queueHead,
                      CL_IN  ClQueueWalkCallbackT fpUserWalkFunction,
                      CL_IN  void* userArg);
```

Parameters:

queueHead: (in) Handle of the queue returned by `clQueueCreate` function.

fpUserWalkFunction: (in) Pointer to the callback function. It accepts the following two parameters:

- ClQueueDataT
- void * - Each element in the queue is passed as the first argument to the callback function.

userArg: (in) User-specified argument. This variable is passed as the second argument to the user's 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 is used to perform a walk on the queue. The user-specified callback function is called with every element (user-data) in the queue.

Library File:
libClUtils

Related Function(s):
None.

3.2.5 clQueueSizeGet

clQueueSizeGet

Synopsis:
Retrieves the number of data elements in the queue.

Header File:
clQueueApi.h

Syntax:

```
ClRcT clQueueSizeGet(
                      CL_IN  ClQueueT queueHead,
                      CL_OUT  ClUint32T* pSize);
```

Parameters:

queueHead: (in) Handle of the queue returned by the `clQueueCreate()` function.

pSize: (out) Pointer to variable of type `ClUint32T`, in which the size of the queue is returned.

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 retrieve the number of data elements in the queue.

Library File:
libClUtils

Related Function(s):
None.

3.2.6 clQueueDelete

clQueueDelete

Synopsis:
Deletes the queue.

Header File:
clQueueApi.h

Syntax:
```
ClRcT clQueueDelete(
                      CL_IN  ClQueueT* pQueueHead);
```

Parameters:
pQueueHead: (in) Pointer to the queue handle returned by `clQueueCreate` function.

Return values:
CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pQueueHead` contains a NULL pointer.

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 delete all the elements in the queue. The destroy callback function, registered during creation is called for every element in the queue.

Library File:
libClUtils

Related Function(s):
clQueueCreate

Chapter 4

Service Management Information Model

TBD

Chapter 5

Service Notifications

TBD

Chapter 6

Debug CLIs

TBD

Index