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

For OpenClovis SDK Release 2.3 V0.4

Document Revision Date: March 08, 2007

Contents

Chapter 1

Functional Overview

The OpenClovis Timer Library enables you to create multiple timers to execute application specific functionality at certain intervals. It performs the following functions:

- Creates multiple timers.
- Specifies a time-out value for each timer.
- Creates one-shot or repetitive timers.
- Specifies an application specific function that should be executed every time the timer fires or expires.

The operating systems such as Linux or BSD have a limit on the number of timers that can be created in an application. However, an application such as OSPF and BGP require a lot of timers to be created at various points during the execution of the application. The Timer Library enables you to create multiple timers that executes a user-defined function when the timer expires. The timer can be of two types: one-shot or repetitive. One-shot timers expire once and are not automatically restarted. Repetitive timers are automatically restarted by the timer library every time they expire.

A user application needs to link to the timer library to utilize it.

The timer library provides APIs to create, delete, start, stop, and restart a timer.

The timer library creates a thread that handles expired timers for each application with which it is linked. As currently implemented, this thread processes expired timers every 10ms.

Chapter 2

Service Model

TBD

Chapter 3

Service APIs

3.1 Type Definitions

3.1.1 ClTimerTimeOutT

typedef struct {
ClUint32T tsMilliSec;
ClUint32T tsSec;
} ClTimerTimeOutT;

The structure `ClTimerTimeOutT` contains the timeout value in seconds and milliseconds.

- *tsMilliSec* - The timeout value in milliseconds.
- *tsSec* - The timeout value in seconds.

3.1.2 ClTimerTypeT

typedef enum {
CL_TIMER_ONE_SHOT=0,
CL_TIMER_REPETITIVE
} ClTimerTypeT;

The enumeration, `ClTimerTypeT`, contains the type of action to be performed on timer expiry. The attributes of the enumeration are:

- *CL_TIMER_ONE_SHOT* - Timer starts automatically after timeout.
- *CL_TIMER_REPETITIVE* - Timer has not started after timeout.

3.1.3 ClTimerContextT

typedef enum {
CL_TIMER_TASK_CONTEXT=0,
CL_TIMER_SEPARATE_CONTEXT
} ClTimerContextT;

The enumeration, `ClTimerContextT`, contains the method of invocation of the timer callback function on timer expiry. The attributes of the enumeration are:

- *CL_TIMER_TASK_CONTEXT* - Timer callback function is called in the same context.
- *CL_TIMER_SEPARATE_CONTEXT* - New thread is created to invoke the callback.

3.1.4 ClTimerCallBackT

*typedef ClRcT(*ClTimerCallBackT)(void *);*

Type of the function that is called on expiration of timer.

3.1.5 ClTimerHandleT

typedef ClHandleT ClTimerHandleT;

The type of the callback function that is called on timer expiry.

3.2 Library Life Cycle functions

3.2.1 clTimerConfigInitialize

clTimerConfigInitialize

Synopsis:
Configures the Timer library.

Header File:
clTimerApi.h

Syntax:

```
ClRcT clTimerConfigInitialize(
          CL_IN void* pConfigData);
```

Parameters:
pConfigData: Pointer to instance of configuration structure. You must pass `ClTimerConfigT` as an input.

Return values:
CL_OK: The function executed successfully.
CL_TIMER_ERR_INVLD_PARAM: An invalid parameter has been passed to the function.
CL_ERR_NULL_POINTER: `pConfigData` contains a NULL pointer.

Description:
This function is used to configure the timer service library. The configurable parameters are:
1. Timer Resolution
This value is in milliseconds and cannot be less than 10 milliseconds. Default value is 10 milliseconds.
2. Timer Task Priority
This value can vary between 1 and 160. Default value is 150.

Library File:
libClTimer

Related Function(s):
None.

3.2.2 clTimerInitialize

clTimerInitialize

Synopsis:
Initializes the Timer library.

Header File:
clTimerApi.h

Syntax:

```
ClRcT clTimerInitialize (void);
```

Parameters:
None.

Return values:
CL_OK: The function executed successfully.

ERROR: Failure initializing the Timer.

Description:
This function is used to initialize the timer service library. After invoking this function the application can invoke other timer related functions.

Library File:
libClTimer

Related Function(s):
clTimerFinalize

3.2.3 clTimerFinalize

clTimerFinalize

Synopsis:
Frees the Timer library.

Header File:
clTimerApi.h

Syntax:
```
ClRcT clTimerFinalize (void);
```

Parameters:
None.

Return values:
CL_OK: The function executed successfully.

ERROR: Failure finalizing the Timer.

Description:
This function is used to free the resources acquired during the initialization of the timer service library. This function is invoked during system shutdown or when the timer is not required. All timers that are currently running are stopped and the data structures required to hold the timer information are deletd.

Library File:
libClTimer

Related Function(s):
clTimerInitialize

3.3 Functional functions

3.3.1 clTimerCreate

clTimerCreate

Synopsis:
Creates a timer.

Header File:
clTimerApi.h

Syntax:

```
ClRcT clTimerCreate(
            CL_IN       ClTimerTimeOutT  timeOut,
            CL_IN       ClTimerTypeT type,
            CL_IN       ClTimerContextT timerTaskSpawn,
            CL_IN       ClTimerCallBackT  fpAction,
            CL_IN       void* pActionArgument,
            CL_INOUT    ClTimerHandleT*   pTimerHandle);
```

Parameters:

timeOut: (in) Timeout value of the timer.

type: (in) Type of the timer to be created. It can be either one-shot or repetitive.

timerTaskSpawn: (in) Determines whether the user-function invoked is in a separate task or in the same context as the timer-task.

fpAction: (in) Function to be called after timer expiry.

pActionArgument: (in) Argument to be passed to the callback function (`fpAction`).

pTimerHandle: (out) Pointer to the memory location where the timer handle created is being copied.

Return values:

CL_OK: The function executed successfully.

CL_TIMER_ERR_INVLD_PARAM: An invalid parameter has been passed to the function.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_NULL_POINTER: `pActionArgument` or `pTimerHandle` contains a NULL pointer.

CL_TIMER_ERR_NULL_CALLBACK_FUNCTION: The callback function passed to this function is NULL.

CL_TIMER_ERR_INVALID_TYPE: The type of the timer is invalid.

CL_TIMER_ERR_INVALID_CONTEXT_TYPE: The context is invalid.

Description:
This function is used to create a new timer. This timer remains inactive until the timer is started. The callback function is executed in the context of the timer task, when the timer expires. This function returns a handle that needs to be specified when you need to start, stop, restart, or destroy the timer.

Library File:
libClTimer

Related Function(s):
clTimerDelete , clTimerStart , clTimerRestart , clTimerCreateAndStart

3.3.2 clTimerDelete

clTimerDelete

Synopsis:
Deletes a timer.

Header File:
clTimerApi.h

Note:
If the timer being deleted is active, then it is made inactive and deleted.

Syntax:

```
ClRcT clTimerDelete ( CL_INOUT ClTimerHandleT* pTimerHandle);
```

Parameters:
pTimerHandle :(in/out) Pointer to timer handle being deleted. The contents are set to zero.

Return values:
CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pTimerHandle` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: `timerHandle` is an invalid handle.

CL_TIMER_ERR_INVALID: The internal timer representation is invalid.

Description:
This function is used to delete an existing timer. It is invoked by the application after the timer has expired. This function is usually called during the application exit, but it can also be called at any other time.

Library File:
libClTimer

Related Function(s):
clTimerCreate , clTimerRestart , clTimerCreateAndStart , clTimerStop

3.3.3 clTimerStart

clTimerStart

Synopsis:
Starts a timer.

Header File:
clTimerApi.h

Syntax:

```
ClRcT clTimerStart (
                        ClTimerHandleT  timerHandle);
```

Parameters:
timerHandle: Handle of the timer being started.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `timerHandle` is an invalid handle.

CL_TIMER_ERR_INVALID: The internal timer representation is invalid.

Description:
This function is used to start a timer. Before the timer can be started, the timer must be created and made active. The callback function is executed when the timeout occurs. The callback function would be executed in the context of the timer task.

Library File:
libClTimer

Related Function(s):
clTimerCreate , clTimerRestart , clTimerCreateAndStart , clTimerStop

3.3.4 clTimerStop

clTimerStop

Synopsis:
Stops a timer.

Header File:
clTimerApi.h

Note:
This function only stops the timer and does not destroy it.

Syntax:

```
ClRcT clTimerStop (
                      CL_IN ClTimerHandleT  timerHandle);
```

Parameters:
timerHandle: (in) Handle of the timer being stopped.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `timerHandle` is an invalid handle.

CL_TIMER_ERR_INVALID: The internal timer representation is invalid.

Description:
This function is used to stop a timer. After invoking this function, the timer becomes inactive, and can be started with `clTimerStart()` or `clTimerRestart()` functions.

Library File:
libClTimer

Related Function(s):
clTimerCreate , clTimerStart , clTimerCreateAndStart , clTimerDelete

3.3.5 clTimerCreateAndStart

clTimerCreateAndStart

Synopsis:
Creates a new timer and activates it.

Header File:
clTimerApi.h

Syntax:

```
ClRcT clTimerCreateAndStart (
            CL_IN           ClTimerTimeOutT  timeOut,
            CL_IN           ClTimerTypeT type,
            CL_IN           ClTimerContextT timerTaskSpawn,
            CL_IN           ClTimerCallBackT  fpAction,
            CL_IN           void  *pActionArgument,
            CL_INOUT        ClTimerHandleT *pTimerHandle);
```

Parameters:

timeOut: (in) Timeout value of the timer.

type: (in) Type of the timer to be created. It can be either one-shot or repetitive.

timerTaskSpawn: (in) Determines if the user-function invoked is in a separate task or in the same context as the timer-task.

fpAction: (in) Function to be called after timer expiry.

actionArgument: (in) Argument to be passed to the callback function (`fpAction`).

pTimerHandle: (in/out) Pointer to the memory location where the timer handle created is being copied.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_PARAMETER: An invalid parameter has been passed to the function.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_ERR_NULL_POINTER: `pActionArgument` or `pTimerHandle` contains a NULL pointer.

CL_TIMER_ERR_NULL_CALLBACK_FUNCTION: An invalid callback function has been passed to the function.

CL_TIMER_ERR_INVALID_TYPE: The type of timer is invalid.

Description:
This function is used to create a new timer and activate it. It is essentially a combination of `tsCreate()` and `tsStart()` functions.
This function is useful when you have to create a new timer and activate it at the time of its creation.

Library File:
libClTimer

Related Function(s):
clTimerCreate , clTimerStart , clTimerStop , clTimerDelete

3.3.6 clTimerRestart

clTimerRestart

Synopsis:
Restarts a timer.

Header File:
clTimerApi.h

Syntax:

```
ClRcT clTimerRestart (
       CL_IN     ClTimerHandleT  timerHandle);
```

Parameters:
timerHandle: (in) Handle of the timer being restarted.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `timerHandle` is an invalid handle.

CL_TIMER_ERR_INVLD_STATE: Timer is in an invalid state.

CL_TIMER_ERR_INVALID: The internal timer representation is invalid.

Description:
This function is used to restart a timer that is created using the `clTimerCreate()`, `clTimerStart()`, or `clTimerCreateAndStart()` functions.

Library File:
libClTimer

Related Function(s):
clTimerCreate, clTimerStart , clTimerCreateAndStart , clTimerStop , clTimerDelete

3.3.7 clTimerUpdate

clTimerUpdate

Synopsis:
Updates a timer.

Header File:
clTimerApi.h

Syntax:

```
ClRcT clTimerUpdate(
            CL_IN               ClTimerHandleT timerHandle,
            CL_IN               ClTimerTimeOutT newTimeout);
```

Parameters:

timerHandle: (in) Handle of the timer being updated.

newTimeout: (in) New timeout value for the timer.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: `timerHandle` is an invalid handle.

CL_TIMER_ERR_INVALID: The internal timer representation is invalid.

Description:
This function is used to update the timeout value of a timer. For this, a timer must be created using `clTimerCreate()` or `clTimerCreateAndStart()` functions. After the timeout value is updated, the timer function is called according to the new timer value.

Library File:
libClTimer

Related Function(s):
clTimerCreate , clTimerStart , clTimerCreateAndStart , clTimerStop , clTimerDelete

3.3.8 clTimerTypeGet

clTimerTypeGet

Synopsis:
Returns the timer type.

Header File:
clTimerApi.h

Syntax:

```
ClRcT clTimerTypeGet (
                CL_IN      ClTimerHandleT timerHandle,
                CL_INOUT   ClUint32T* pTimerType);
```

Parameters:

timerHandle: (in) Handle of the timer.

pTimerType: (in/out) The pointer to the location to which the type of the timer is being copied. It can have two values: If the value is:

- 0: The timer type is one-shot.
- 1: The timer type is repetitive.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pTimerType` contains a NULL pointer.

CL_ERR_INVALID_HANDLE: `timerHandle` is an invalid handle.

CL_TIMER_ERR_INVALID: The internal timer representation is invalid.

Description:
This function is used to query and return the type of timer, one-shot or repetitive. A one-shot timer is invoked only once and then the timer stops operating. For a repetitive timer, the timeout occurs periodically until the timer is stopped.

Library File:
libClTimer

Related Function(s):
clTimerCreate , clTimerStart , clTimerCreateAndStart , clTimerStop , clTimerDelete

Chapter 4

Service Management Information Model

TBD

Chapter 5

Service Notifications

TBD

Chapter 6

Debug CLIs

TBD

Index