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

For OpenClovis SDK Release 2.3 V0.4

Document Revision Date: March 27, 2007

Contents

Chapter 1

Functional Overview

The OpenClovis Event Service is a communication mechanism based on the concept of event channels. It is a publish and/or subscribe mechanism where a publisher can communicate with one or more subscribers asynchronously by publishing events over an event channel. The publishers and subscribers can be any component residing anywhere in the cluster.

An Event has a standard header and zero or more bytes of publisher event data. Publishers and Subscribers can communicate over multiple event channels. Multiple publishers and multiple subscribers can communicate over a single event channel. Subscribers are anonymous, which means that they can join and leave an event channel at any time without involving the publisher(s). An Event publisher can also be an Event subscriber.

The Event Service function does not impose a specific layout or format for the published event data. Publishers and subscribers on an event channel must agree on the structure of the data for events published on that event channel. They can use data marshalling, if support for heterogeneity is required. Conventions on the structure of the event data can vary from one event channel to another.

Chapter 2

Service Model

2.1 Usage Model

A component can subscribe to receive events on an event channel using the `clEventSubscribe()` function. The Event Service delivers events to a subscribing component using the `clEvtEventDeliverCallback()` function of that component. To stop receiving events for which it has registered, a subscriber can invoke the `clEventUnsubscribe()` function. If a component terminates abnormally, the Event Service closes all of its open event channels.

2.2 Functional Description

2.2.1 Events

An event consists of a standard set of event attributes (event header) and zero or more bytes of event data. An event header is allocated using the `clEventAllocate()` function and is released using the `clEventFree()` function. The `clEventAllocate()` function returns a handle that can be used in subsequent invocations of the Event Service functions. The event attributes are written and read using `SET` and `GET` functions of the Event Service function. These attributes cannot be read and written directly. An event is published using the `clEventPublish()` function. The parameters to be specified for this function are event handle, additional information (optional), and event data contained in a data buffer. Thus, a published event consists of the event header that contains the set of attributes and additional information (optional) contained in the data buffer.

2.2.2 Event Channels

An event channel enables multiple publishers to communicate with multiple subscribers. It is global to a cluster and is identified by a unique name. To use the Event Service, a component must open an event channel using the `clEventChannelOpen()` or `clEventChannelOpenAsync()` function. The event channel is created, if it does not exist. A component can open an event channel to publish events and subscribe to receive events. Publishers can also act as subscribers on the same event channel. Event channels can be deleted using the `clEventChannelUnlink()` function. When an event is allocated for an

event channel using the `clEventAllocate()` call, it can be published several times on the same event channel, by changing its attributes prior to each publication.

The OpenClovis Event Channel has the following features:

- Best effort delivery: The Event Service provides best effort delivery of events to an anonymous set of subscribers. A published event can be lost or it can be delivered to a subset of the subscribers. Some subscribers receive the event while others do not. For example, there is no guarantee that an event is delivered to all existing subscribers, if the publisher fails while publishing the event. A subscriber may lose events, if the subscriber node is overwhelmed with events.
- At most once delivery: The Event Service does not deliver the same event for a subscription of a subscriber multiple times.
- Event priority: Events are published with a certain priority. High priority events are delivered ahead of low priority events.
- Event ordering: For a given priority, events are received by subscribers in the order in which the publisher published the events.
- Retention time: Events published with a non-zero retention period are retained for a specified duration. This duration can be provided while allocating an Event. This provides the opportunity for new subscribers to obtain events that were published before their subscription to the event channel. Processes can use the Event Service functions to remove events before the retention time expires.
- Payload structure: The Event Service function does not impose a specific layout or format for the published event data or payload. Publishers and subscribers on an event channel must agree on the structure of the data for events published on that event channel and can use data marshalling support for heterogeneity is required. Conventions on the structure of the event data can vary from one event channel to another.
- Event Filtering: The standard set of event attributes include an array of event patterns. The values of these patterns are set by the event publisher and are used to organize events into various categories. All users (publishers and subscribers) of an event channel must share the same conventions regarding the number of patterns being used, their ordering, contents, and meaning. For example, an event channel, used to notify changes made to a relational database, can define events where only three patterns are used as follows:

  1. The first pattern contains the name of the database being modified.
  2. The second pattern contains the name of the table being modified.
  3. The third pattern contains the primary key of the record being modified.

  The event data can be used to provide a description of the modified fields and the old/new values. Event patterns play an important role in the Event Service, as they are the basis for filtering the events that must be delivered to a particular subscriber. When subscribing on an event channel, a component must specify the filters that need to be applied on published events. Only events which satisfy the filters are delivered to the component. Using the previous example of the database notifications published on an event channel, a subscriber can provide a filter array indicating:

  - The name of a database required by the subscriber.
  - The name of a table required by the subscriber.
  - No filter for the primary key. If a filter is not specified, the component will receive all notification events related to the specified table, in the specified database for any primary key.

Chapter 3

Service APIs

3.1 Type Definitions

3.1.1 ClEventInitHandleT

typedef ClHandleT ClEventInitHandleT;

The type of the handle provided by the Event Service to a component during the initialization of the Event Service library. It is used by the component when it invokes the Event Service function, so that the component can be recognized by the Event Service.

3.1.2 ClEventChannelHandleT

typedef ClHandleT ClEventChannelHandleT;

The type of the handle of an event channel. This is provided by the Event Service to a component that has opened a channel for publishing or receiving events. The component has to use this handle for using the Event service functions such as `clEventSubscribe()`, `clEventAllocate()`, and so on as this handle enables the Event service associate subscriptions with an Event channel.

3.1.3 ClEventHandleT

typedef ClHandleT ClEventHandleT;

The type of the handle to an event provided by the Event Service, to a component that needs to publish an event or process a received event. This handle can be used by the component to use the Event functions such as `clEventDataGet()`, `clEventPublish()`, and so on.

3.1.4 ClEventCallbacksT

typedef struct {
ClEventChannelOpenCallbackT clEvtChannelOpenCallback;
ClEventDeliverCallbackT clEvtEventDeliverCallback;

} ClEventCallbacksT;

The type of the callback structure supplied by a component to the Event Service containing the callback functions that can be invoked by the Event Service.

- *clEvtChannelOpenCallback* - An asynchronous channel open callback. This is invoked when the `clEventChannelOpenAsync()` call, by the component to open a channel is completed.
- *clEvtEventDeliverCallback* - This callback is executed when an event required (subscribed) by a component, is delivered to the component.

3.1.5 ClEventChannelOpenFlagsT

typedef ClUint8T ClEventChannelOpenFlagsT;

The type of the event Channel open flag. It is named as an open flag as it is used to open a channel (`clEventChannelOpen()`). This flag is used to inform the Event service if the user is a subscriber, publisher or both. It also defines the scope for the channel. It specifies if the event channel is local or global. The values of this flag are:

- *CL_EVENT_LOCAL_CHANNEL*
- *CL_EVENT_GLOBAL_CHANNEL*
- *CL_EVENT_CHANNEL_PUBLISHER*
- *CL_EVENT_CHANNEL_SUBSCRIBER*
- *CL_EVENT_CHANNEL_CREATE*

The value of this flag is set by performing a bitwise OR with one or more of the following:

- *CL_EVENT_CHANNEL_PUBLISHER* - To enable the component to use the event channel handle returned by the `clEventPublish()` function.
- *CL_EVENT_CHANNEL_SUBSCRIBER* - To enable the component to use the event channel handle returned by the `clEventSubscribe()` function.
- *CL_EVENT_CHANNEL_CREATE* - To open and create an event channel that does not exist.
- *CL_EVENT_LOCAL_CHANNEL* and *CL_EVENT_GLOBAL_CHANNEL* - Flags that specify the scope of the event channel. `CL_EVENT_LOCAL_CHANNEL` and `CL_EVENT_GLOBAL_CHANNEL` cannot be ORed with each other as they cannot be used simultaneously.

3.1.6 ClEventPatternArrayT

typedef struct {
ClSizeT allocatedNumber;
ClSizeT patternsNumber;
*ClEventPatternT *pPatterns;*
} ClEventPatternArrayT;

The structure, `ClEventPatternArrayT`, defines the type of an event pattern array. Its attributes are:

- *allocatedNumber* - Number of entries allocated in the pattern buffer.
- *patternsNumber* - Number of patterns in the event.
- *pPatterns* - Pointer to a buffer where the array of pattern is copied.

3.1.7 ClEventPriorityT

typedef ClUint8T ClEventPriorityT;

Every event has a priority associated with it. This priority controls the sending and delivery of the order of events. The priority ranges from: `CL_EVENT_LOWEST_PRIORITY` to `CL_EVENT_HIGHEST_PRIORITY`.

3.1.8 ClEventIdT

typedef ClUint64T ClEventIdT;

The type of an event identifier. Values ranging from 0 to 1000 have special meanings and cannot be used by the event service to identify regular events.

3.1.9 ClEventSubscriptionIdT

typedef ClUint32T ClEventSubscriptionIdT;

The type of an identifier for a subscription by a component on an event channel. This identifier is used to associate the delivery of events for that subscription with the component. Subscription IDs are unique for every subscription.

3.1.10 ClEventFilterArrayT

```
typedef struct {
        ClSizeT filtersNumber;
        ClEventFilterT *pFilters;
} ClEventFilterArrayT;
```

The structure, `ClEventFilterArrayT`, defines a set of filters. Filters are passed to the Event Service by a subscriber component through the `clEvtEventSubscribe()` call. The Event Service performs the filtering operation to decide if a published event is to be delivered to a subscriber for a given subscription. The filtering is performed by matching the first filter (contents and type) against the first pattern in the event pattern array, the second filter against the second pattern in the event pattern array, and so on till the last filter is reached. An event matches a given subscription, if the patterns of the event match all filters provided in the `clEvtEventSubscribe()` function call.

The attributes of the structure are:

- *filtersNumber* - Number of filters.
- *pFilters* - Pointer to filter pattern.

3.2 Library Life Cycle APIs

3.2.1 clEventInitialize

clEventInitialize

Synopsis:
Initializes the Event Service for the calling component and ensures version compatibility.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventInitialize(
            CL_OUT ClEventInitHandleT* pEvtHandle,
             CL_IN const ClEventCallbacksT* pEvtCallbacks,
            CL_INOUT ClVersionT* pVersion );
```

Parameters:

pEvtHandle: (out) Pointer to the handle that identifies this initialization of the Event Service. This parameter is returned by this function.

pEvtCallbacks: (in) If `evtCallbacks` is set to NULL, a callback is not registered. Otherwise, it acts as a pointer to the `ClEventCallbacksT` structure, that contains the callback functions of the component that the Event Service can use. Only callback functions that are not NULL, are registered in this structure.

pVersion: (in/out) As an input parameter, `version` is a pointer to the required Event Service version. `minorVersion` is ignored and should be set to `0x00`. As an output parameter, the version supported by Event Service is delivered.

Return values:

CL_OK: The Event service initialization was successful.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred within the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_ERR_VERSION_MISMATCH: The provided version is not supported by the current implementation of the Event Service.

CL_ERR_NULL_POINTER: `pEvtHandle` or `pVersion` contains a NULL pointer.

CL_ERR_NO_MEMORY: There is no memory available for the Event service or a module of the Event service. Thus, the service cannot be provided and this can be a transient problem.

CL_ERR_NO_RESOURCE: There is no resource (other than memory) available for the Event service or a module of the Event service. Thus, the service cannot be provided and this can be a transient problem.

Description:
This function is used to initialize the Event Service for the invoking component and registers the various callback functions. It must be invoked prior to the invocation of any other Event Service functionality. The handle, `pEvtHandle`, is returned as a reference to this association between the component and the Event Service. The component uses this handle in subsequent communication with the Event Service.

If the implementation supports the required releaseCode, and a majorVersion is greater than or equal to the required majorVersion, CL_AIS_OK is returned.
If the implementation supports the required releaseCode (a member of pVersion), and its majorVersion is greater than or equal to the required majorVersion, CL_OK is returned and pVersion is set to:

- releaseCode = Required releaseCode.
- majorVersion = Highest major version supported for required releaseCode
- minorVersion = Highest minor version supported for the returned releaseCode and majorVersion

If the mentioned condition is not satisfied, CL_ERR_VERSION_MISMATCH is returned and pVersion is set to:

- releaseCode: Required releaseCode, if it is supported
- Lowest releaseCode higher than required releaseCode, if the required releaseCode is lower than any supported releaseCode
- Highest releaseCode lower than required releaseCode, if the required releaseCode is higher than any supported releaseCode
- majorVersion = Highest major version supported for returned releaseCode
- minorVersion = Highest minor version supported for returned releaseCode and majorVersion

Library File:
ClEventClient

Related Function(s):
clEventSelectionObjectGet, clEventDispatch, clEventFinalize

3.2.2 clEventSelectionObjectGet

clEventSelectionObjectGet

Synopsis:
This function is used to retrieve the operating system handle, `selectionObject`, associated with the handle, `evtHandle`.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventSelectionObjectGet(
            CL_IN ClEventInitHandleT evtHandle,
              CL_OUT ClSelectionObjectT *pSelectionObject);
```

Parameters:

evtHandle: (in) The handle that identifies this initialization of the Event Service, obtained using the `clEventInitialize()` function.

pSelectionObject: (out) A pointer to the operating system handle that the invoking component can use to detect pending callbacks.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `evtHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_ERR_NO_MEMORY: There is no memory available for the Event service or a module of the Event service. Thus, the service cannot be provided and this can be a transient problem.

CL_ERR_NO_RESOURCE: There is no resource (other than memory) available for the Event service or a module of the Event service. Thus, the service cannot be provided and this can be a transient problem.

Description:
This function returns the operating system handle, `pSelectionObject`, associated with the handle `evtHandle`. A component can use this handle to detect pending callbacks, instead of repeatedly calling the `clEventDispatch()` function.
`pSelectionObject`, is a file descriptor that can be used with `poll()` or `select()` system calls to detect incoming callbacks.
`selectionObject`, becomes invalid when `clEventFinalize()` is executed on the same handle, `evtHandle`.

Library File:
ClEventClient

Related Function(s):
clEventInitialize, clEventDispatch, clEventFinalize

3.2.3 clEventDispatch

clEventDispatch

Synopsis:
Invokes the pending callbacks in the context of the calling component.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventDispatch(
            CL_IN ClEventInitHandleT evtHandle,
             CL_IN ClDispatchFlagsT dispatchFlags);
```

Parameters:

evtHandle: (in) The handle (obtained through the `clEventInitialize()`) that identifies this initialization of the Event service.

dispatchFlags: (in) Flags that specify how the callbacks of the `clEventDispatch()` function must be executed. This parameter can accept the values, `CL_DISPATCH_ONE`, `CL_DISPATCH_ALL`, or `CL_DISPATCH_BLOCKING`.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `evtHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_ERR_NO_MEMORY: There is no memory available for the Event service or a module of the Event service. Thus, the service cannot be provided and this can be a transient problem.

CL_ERR_NO_RESOURCE: There is no resource (other than memory) available for the Event service or a module of the Event service. Thus, the service cannot be provided and this can be a transient problem.

Description:
This function invokes pending callbacks for `evtHandle` (in the context of the calling component), as specified in the `dispatchFlags` parameter.

Library File:
ClEventClient

Related Function(s):
clEventInitialize, clEventSelectionObjectGet

3.2.4 clEventFinalize

clEventFinalize

Synopsis:
Finalizes the Event service library.

Header File:
clEventApi.h

Syntax:
```
ClRcT clEventFinalize(
            CL_IN ClEventInitHandleT evtHandle );
```

Parameters:
evtHandle: (in) The handle (obtained through the `clEventInitialize()`) that identifies this initialization of the Event service.

Return values:
CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `evtHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

Description:
This function is used to close the association, identified by evtHandle, between the invoking component and Event Service. The component must call the `clEventInitialize()` function before using this function. If the `clEventFinalize()` function is executed successfully, all resources acquired by the `clEventInitialize()` function are released. It also removes the open event channels and cancels the pending callbacks related to the handle.
As the callback invocation is asynchronous, some callbacks can still be executed after this function is called.

Library File:
ClEventClient

Related Function(s):
clEventInitialize

3.3 Functional APIs

3.3.1 clEventChannelOpen

clEventChannelOpen

Synopsis:
Opens an event channel for the requesting component that needs to publish and/or subscribe to events.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventChannelOpen(
            CL_IN ClEventInitHandleT evtHandle,
            CL_IN const ClNameT *pEvtChannelName,
            CL_IN ClEventChannelOpenFlagsT evtChannelOpenFlag,
            CL_IN ClTimeT timeout,
            CL_OUT ClEventChannelHandleT *pChannelHandle );
```

Parameters:

evtHandle: (in) The handle obtained by the `clEventInitialize()` function designating this particular initialization of the Event Service.

pEvtChannelName: (in) Pointer to the name of the event channel that identifies the event channel.

evtChannelOpenFlags: (in) The requested access modes and scope of the event channel. For more information, refer to `evtChannelOpenFlagT` in the Type Definitions section.

timeout: (in) Time-out for calling this function.

pChannelHandle: (out) Pointer to the handle of the event channel, provided by the invoking component in the address space of the component. If the event channel is opened successfully, the Event Service stores the handle in `pChannelHandle`. This handle is used by the component to access the channel in subsequent invocations of the Event Service functions.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `evtHandle` or `pChannelHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function

CL_ERR_NO_MEMORY: The Event service library or a module of the Event service is out of memory. The service cannot be provided at this time. This can be a transient problem.

CL_ERR_NO_RESOURCE: The Event Service library or a module of Event service is out of resources (other than memory). The service cannot be provided at this time. This can be a transient problem.

CL_EVENT_ERR_CHANNEL_ALREADY_OPENED: The event channel is already opened by a component with the same name and scope.

Description:

This function opens an event channel. If the event channel does not exist and `CL_EVENT_CHANNEL_CREATE` flag is set in `channelOpenFlags`, the event channel is created. This function is a blocking operation and returns a new event channel handle.

Note:

An event channel can be created even after the time-out period expires.

Library File:

ClEventClient

Related Function(s):

clEventInitialize, clEventChannelClose

3.3.2 clEventChannelOpenAsync

clEventChannelOpenAsync

Synopsis:
Opens an event channel asynchronously.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventChannelOpenAsync(
            CL_IN ClEventInitHandleT evtHandle,
            CL_IN ClInvocationT invocation,
            CL_IN const ClNameT *pEvtChannelName,
            CL_IN ClEventChannelOpenFlagsT channelOpenFlags );
```

Parameters:

evtHandle: (in) Handle obtained by the `clEventInitialize()` function designating this particular initialization of the Event Service.

invocation: (in) Allows the invoking component to match the invocation of `clEventChannelOpenAsync()` function with the corresponding callback.

pEvtChannelName: (in) Pointer to the name of the event channel that identifies the event channel.

evtChannelOpenFlags: (in) The requested access modes and scope of the event channel. For details, refer `evtChannelOpenFlagT` in the Type Definitions section.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_ERR_NO_MEMORY: The Event service library or a module of the Event service is out of memory. The service cannot be provided at this time. This can be a transient problem.

CL_ERR_NO_RESOURCE: The Event Service library or a module of Event service is out of resources (other than memory). The service cannot be provided at this time. This can be a transient problem.

CL_EVENT_ERR_CHANNEL_ALREADY_OPENED: The event channel is already opened by a component with the same name and scope.

Description:

This function opens an event channel asynchronously. The event channel is created, if the event channel does not exist and the `CL_EVENT_CHANNEL_CREATE` flag is set in `channelOpenFlags`. The `clEvtChannelOpenCallback()` function is invoked when this function is successfully executed. The component provides the `invocation` value to this function and the Event Service provides it to the calling component using the `clEvtChannelOpenCallback()` function.

Library File:
ClEventClient

Related Function(s):
clEvtChannelOpenCallbackT(), clEventInitialize, clEventChannelClose

3.3.3 clEventChannelClose

clEventChannelClose

Synopsis:
Closes an event channel.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventChannelClose(
            CL_IN ClEventChannelHandleT channelHandle );
```

Parameters:
channelHandle: (in) Handle of the event channel to be closed, obtained using the `clEventChannelOpen()` or `clEvtChannelOpenCallback()` functions.

Return values:
CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `channelHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

Description:
This function closes the event channel identified by `channelHandle`. After the execution of this function, the `channelHandle` becomes invalid.

Library File:
ClEventClient

Related Function(s):
clEventChannelOpen, clEventChannelOpenAsync, clEvtChannelOpenCallbackT, clEventChannelUnlink

3.3.4 clEventChannelUnlink

clEventChannelUnlink

Synopsis:
Deletes an event channel.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventChannelUnlink(
            CL_IN ClEventInitHandleT evtHandle,
            CL_IN const ClNameT *pEvtChannelName );
```

Parameters:

evtHandle: (in) Handle obtained by the invocation of `clEventInitialize()` designating this particular initialization of the Event Service.

pEvtChannelName: (in) Pointer to the name of the event channel to be unlinked.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `evtHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

Description:
This function deletes an existing event channel, identified by `pEvtChannelName`, from the cluster. In the current implementation of the event service, this functionality of unlink is not implemented and a call to `clEventChannelUnlink` returns `CL_OK`, unconditionally.

Library File:
ClEventClient

Related Function(s):
clEventInitialize, clEventChannelClose

3.3.5 clEventAllocate

clEventAllocate

Synopsis:
This function allocates an event header.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventAllocate(
            CL_IN ClEventChannelHandleT channelHandle,
            CL_OUT ClEventHandleT* pEventHandle );
```

Parameters:

channelHandle: (in) Handle of the event channel on which the event is to be published. It must be obtained earlier, either using the `clEventChannelOpen()` function or `clEvtChannelOpenCallback()` function.

pEventHandle: (out) Pointer to the handle of the newly allocated event. The calling component must allocate memory for this handle before this function is executed. The Event Service assigns the value of the `eventHandle` when this function is invoked.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `channelHandle` or `pEventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_NO_MEvent Service: Memory allocation failure.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_ERR_NO_MEMORY: The Event service library or a module of the Event service is out of memory. The service cannot be provided at this time. This can be a transient problem.

CL_ERR_NO_RESOURCE: The Event Service library or a module of Event service is out of resources (other than memory). The service cannot be provided at this time. This can be a transient problem.

CL_EVENT_ERR_NOT_OPENED_FOR_PUBLISH: : The flag, `CL_EVENT_CHANNEL_PUBLISHER`, was not set in the `ClEventChannelOpenFlagsT`, while opening the event channel on which this event was allocated. `ClEventAllocate` does not allow such a component to allocate an event on that channel.

Description:
This function is used to allocate memory for the event header. The event allocated by this function must be freed using the `clEventFree()` function.

Library File:
ClEventClient

Related Function(s):
clEventFree, clEventPublish()

3.3.6 clEventFree

clEventFree

Synopsis:
Frees an event header that is allocated using `clEventAllocate()`.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventFree(
            CL_IN ClEventHandleT eventHandle );
```

Parameters:
eventHandle: (in) Handle of the event for which memory is to be freed by the Event Service.

Return values:
CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

Description:
This function is used to give permission to the Event Service to de-allocate the event associated with the `eventHandle` including the memory containing the attributes and the event data (if present). It frees the events allocated by `clEventAllocate()` function or `clEvtEventDeliverCallback()` function.

Library File:
ClEventClient

Related Function(s):
clEventAllocate, clEventChannelOpen, clEventChannelOpenAsync

3.4 Event APIs

3.4.1 clEventAttributesSet

clEventAttributesSet

Synopsis:
This function is used to set (assign values) the attributes of an event.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventAttributesSet(
            CL_IN ClEventHandleT eventHandle,
            CL_IN const ClEventPatternArrayT *pPatternArray,
            CL_IN ClEventPriorityT priority,
            CL_IN ClTimeT retentionTime,
            CL_IN const ClNameT *pPublisherName );
```

Parameters:
eventHandle: (in) Handle of the event for which attributes are to be set.
pPatternArray: (in) Pointer to a structure that contains the array of patterns. For details, refer to the `ClEventPatternArrayT` in the Type Definitions section.
priority: (in) Priority of the event.
retentionTime: (in) Duration for which the event is retained.
pPublisherName: (in) Pointer to the name of the event publisher.

Return values:
CL_OK: The function executed successfully.
CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.
CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.
CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.
CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.
CL_ERR_NO_MEMORY: The Event service library or a module of the Event service is out of memory. The service cannot be provided at this time. This can be a transient problem.
CL_ERR_NO_RESOURCE: There is no resource (other than memory) available for the Event service or a module of the Event service. Thus, the service cannot be provided and this can be a transient problem.

Description:
This function is used to set all the writeable event attributes identified by eventHandle. These attributes are: `pPatternArray`, `priority`, `retentionTime`, and `pPublisherName`. If `pPatternArray` or `pPublisherName` is NULL, the corresponding attributes are not changed.

Library File:
ClEventClient

Related function(s):
clEventAllocate, clEventFree, clEvtEventDeliverCallbackT, clEventAttributesGet, clEventChannelOpen, clEventChannelOpenAsync

3.4.2 clEventAttributesGet

clEventAttributesGet

Synopsis:

Returns the attributes of an event. It provides a component with the priority, retention time, publisher name, publish time, and event identifier information for an event.

Header File:

clEventApi.h

Syntax:

```
ClRcT clEventAttributesGet(
            CL_IN ClEventHandleT eventHandle,
            CL_IN ClEventPatternArrayT *pPatternArray,
            CL_OUT ClEventPriorityT *pPriority,
            CL_OUT ClTimeT *pRetentionTime,
            CL_OUT ClNameT *pPublisherName,
            CL_OUT ClTimeT *pPublishTime,
            CL_OUT ClEventIdT *pEventId );
```

Parameters:

eventHandle: (in) Handle of the event whose attributes are to be retrieved.

pPatternArray: (in) Pointer to a structure, `ClEventPatternArrayT`, that describes the event pattern array and the number of patterns to be retrieved. The attributes of this structure are:

- *allocatedNumber* - Number of entries allocated in the pattern buffer.
- *patternsNumber* - Actual number of patterns in the event.
- *pPatterns* - Pointer to a buffer where the array of pattern is copied.
- If ppatterns is set to NULL, Event Service ignores the allocatedNumber attribute, allocates memory for `pPattern` array and the individual patterns, and sets `allocatedNumber`, `patternsNumber`, and `pPatterns` accordingly.
- The calling component is responsible for de-allocating the corresponding memory for each element of the `pPatterns` array.
- Alternatively, the invoking component can allocate memory to retrieve all event patterns and set the fields, `allocatedNumber`, `patternsNumber`, and `pPatterns` accordingly.
- These fields are `input` parameters and are not modified by the Event Service. The Event Service copies the patterns into successive entries of patterns, starting with the first entry and continuing till all event patterns are copied.
- If `allocatedNumber` is smaller than the number of event patterns or, if the size of the buffer allocated for one of the patterns is less then the actual size of the pattern, the invocation fails and the error, `CL_EVENT_ERR_NO_SPACE` is returned.
- Regardless of whether such an error occurs, the Event Service sets the `pPatternArray>patternsNumber` and `pPatternArray>patterns[i].patternSize` fields for all `pPatternArray>allocatedNumber` individual patterns, to indicate the number of event patterns and the size of each pattern.

pPriority: (out) Pointer to the priority of the event.

pRetentionTime: (out) Pointer to the time duration for which the publisher retains the event.

pPublisherName: (out) Pointer to the name of the event publisher.

pPublishTime: (out) Pointer to the time when the publisher published the event.

pEventId: (out) Pointer to the event identifier.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_EVENT_ERR_NO_SPACE: Buffer provided by the component is not sufficient to hold the data associated with the delivered event.

Description:

This function is used to retrieve the value of the attributes of the event identified by eventHandle. If the invoking component provides a NULL reference for each of the `out` or `in/out` parameters, the Event Service does not return the `out` value. This function can be called on any event allocated by the `clEventAllocate()` function or received through the `clEvtEventDeliverCallback()` function. This can also be modified by the `clEventAttributesSet()`. If this function is invoked on a received event, the attributes `publish time` and `event id` retain the values set by the Event Service at the time the event was published. Otherwise, the attributes will either have the initial values set by the Event Service while allocating the event or the attributes set by a call to `clEventAttributesSet` function.

Library File:

ClEventClient

Related Function(s):

clEventAllocate, clEventFree, clEventChannelOpen, clEventChannelOpenAsync, clEventAttributesSet

3.4.3 clEventDataGet

clEventDataGet

Synopsis:
Retrieves the data associated with an event.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventDataGet(
            CL_IN ClEventHandleT eventHandle,
            CL_INOUT void *pEventData,
            CL_INOUT ClSizeT *pEventDataSize );
```

Parameters:

eventHandle: (in) Handle to the event delivered by the `clEvtEventDeliverCallback()` function.

pEventData: (in/out) Pointer to a buffer provided by component in which Event Service stores the data associated with the delivered event is stored. If `pEventData` is NULL, the value of `pEventDataSize` (provided by the invoking component) is ignored and the buffer is provided by the Event Service library. The buffer must be de-allocated by the calling component after returning from the `clEventDataGet()` call.

pEventDataSize: (in/out) If `pEventData` is not NULL, the `in` value of this parameter is same as the `pEventData` buffer provided by the invoking component. If the size of the buffer is smaller than the size of the data, the data is not copied into the buffer and the error `CL_EVENT_ERR_NO_SPACE` is returned. If pEventData is NULL, the `in` value of `pEventDataSize` is ignored. The `out` value of `pEventDataSize` is SET when the function returns either `CL_AIS_OK` or `CL_EVENT_ERR_NO_SPACE`, and its size is equal to the data associated with this event.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_EVENT_ERR_NO_SPACE: Buffer provided by the component is not sufficient to hold the data associated with the delivered event.

Description:
This function is used to retrieve the data associated with an event earlier delivered by the `clEvtEventDeliverCallback()` function.

Library File:
ClEventClient

Related Function(s):
clEvtEventDeliverCallbackT, clEventFree, clEventChannelOpen, clEventChannelOpenAsync,

3.4.4 clEventCookieGet

clEventCookieGet

Synopsis:
Returns the cookie previously passed to the `clEventSubscribe()` function.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventCookieGet(
            CL_IN ClEventHandleT eventHandle,
             CL_OUT void** ppCookie);
```

Parameters:

eventHandle: (in) Handle to the event delivered by `clEvtEventDeliverCallback()` function.

ppCookie: (out) Cookie, required for subscribing to an event.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

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

Description:
This function is used to return the cookie passed by the component when it subscribes to an event. It can be invoked from the registered callback function.

Library File:
ClEventClient

Related Function(s):
clEventSubscribe

3.4.5 clEventPublish

clEventPublish

Synopsis:
Publishes an event on the event channel.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventPublish(
            CL_IN ClEventHandleT eventHandle,
            CL_IN const void *pEventData,
            CL_IN ClSizeT eventDataSize,
            CL_OUT ClEventIdT *pEventId );
```

Parameters:

eventHandle: (in) Handle of the event that needs to be published. The event must be allocated by the `clEventAllocate()` function or obtained using the `clEvtEventDeliverCallback()` function. The patterns must be set by `clEventAttributesSet()` function, if changes are required.

pEventData: (in) Pointer to a buffer that contains additional information about the event being published. This parameter is set to NULL, if no additional information is associated with the event. The component can de-allocate the memory for `pEventData` when the `clEventPublish()` function returns.

eventDataSize: (in) The number of bytes in the buffer pointed to by `pEventData`. This parameter is ignored, if `pEventData` is NULL.

pEventId: (out) Pointer to the identifier of the event.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_EVENT_ERR_NOT_OPENED_FOR_PUBLISH: The flag, `CL_EVENT_CHANNEL_PUBLISHER`, was not set when the channel was opened. The publisher did not open the channel as a publisher.

Description:

This function is used to publish an event on the channel for which the event specified by `eventHandle` has been allocated or obtained through the `clEvtEventDeliverCallback()` function. It returns the event identifier in `eventId`. The event to be published consists of a standard set of attributes, such as the event header, and optional data. To publish events, the publisher must open the channel as a publisher. This can be done by setting the `CL_EVENT_CHANNEL_PUBLISHER` flag. Before an event can be published, the publisher component can call the `clEventAttributesSet()` function to set the writeable event attributes. The published event is delivered to the subscribers whose subscription filters match the event patterns. When the Event Service publishes an event, it automatically sets the following read-only event attributes into the published event:

- Event publish time
- Event identifier

In addition to the event attributes, a component also provides values for the `pEventData` and `eventDataSize` parameters for publication as part of the event. The event attributes and the event data of the event, `eventHandle`, are not affected by this function. This function copies the event attributes and the event data into the internal memory of the Event Service. The invoking component can free the event using `clEventFree()`, after `clEventPublish()` returns successfully.

Library File:
ClEventClient

Related Function(s):
clEventAllocate, clEventFree, clEvtEventDeliverCallbackT, clEventAttributesSet, clEventSubscribe

3.4.6 clEventSubscribe

clEventSubscribe

Synopsis:
Subscribes to an event identified by an event type (filter).

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventSubscribe(
            CL_IN ClEventChannelHandleT channelHandle,
            CL_IN const ClEventFilterArrayT *pFilters,
            CL_IN ClEventSubscriptionIdT subscriptionId,
            CL_IN void* pCookie);
```

Parameters:

channelHandle: (in) Handle of the event channel, on which the component is subscribing, to receive events. The parameter, `channelHandle`, must be obtained using the `clEventChannelOpen()` or `clEvtChannelOpenCallback()` functions.

pFilters: (in) Pointer to `ClEventFilterArrayT` structure, allocated by the component, that defines the filter patterns to filter events received on the event channel. The component can de-allocate memory for the filters when `clEventSubscribe()` returns.

pSubscriptionId: (in) Identifies a specific subscription on this instance of the opened event channel corresponding to the `channelHandle` that is used as a parameter to the `clEvtEventDeliverCallback()` function.

pCookie: (in) Cookie, that is required to be provided by the component. It can be retrieved using the `clEvtEventUtilsCookieGet()` function.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_EVENT_ERR_NOT_OPENED_FOR_SUBSCRIPTION: The flag, `CL_EVENT_CHANNEL_PUBLISHER`, was not set when the channel was opened. The publisher did not open the channel as a publisher.

Description:

This function enables a component to subscribe for events on an event channel by registering one or more filters on that event channel. Events are delivered using the `clEvtEventDeliverCallback()` function supplied when the component invokes the `clEventInitialize()` function. The component must open the event channel, `channelHandle`, with the `CL_EVENT_CHANNEL_SUBSCRIBER` flag set for the successful execution of this function. The memory associated with the filters is not de-allocated by the `clEventSubscribe()` function. It is the responsibility of the invoking component to de-allocate the memory when the `clEventSubscribe()` function returns. For a given subscription, the `pFilters` parameter cannot be modified. To change the filters parameter

without losing events, a component must establish a new subscription (or subscribe again) by assigning new values to the `pFilters` parameter. The old subscription can be removed using the `clEventUnsubscribe()` function when the new subscription is established.

Library File:
ClEventClient

Related Function(s):
clEventUnsubscribe, clEventDataGet, clEvtEventDeliverCallbackT, clEventAttributesGet

3.4.7 clEventUnsubscribe

clEventUnsubscribe

Synopsis:
Cancels the subscription to an event that was subscribed earlier.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventUnsubscribe(
            CL_IN ClEventChannelHandleT channelHandle,
            CL_IN ClEventSubscriptionIdT subscriptionId );
```

Parameters:

channelHandle: (in) Event channel for which the subscription needs to be deleted. This request for cancellation of the subscription is made by the subscriber to the Event Service. This handle must be obtained using the `clEventChannelOpen()` or `clEvtChannelOpenCallback()` functions.

subscriptionId: (in) Identifier of the subscription.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `channelHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_ERR_NO_MEMORY: There is no memory available for the Event service or a module of the Event service. Thus, the service cannot be provided and this can be a transient problem.

Description:
This function enables a component to stop receiving events for a particular subscription on an event channel by removing the subscription. The execution of this function is successful if the subscription ID matches with an earlier registered subscription. Events queued to be delivered to the component and those that do not match any subscription in the `clEventUnsubscribe()` function are purged. A component that needs to modify a subscription without losing any events must establish the new subscription before removing the existing subscription.

Library File:
ClEventClient

Related Function(s):
clEventSubscribe

3.4.8 clEventRetentionTimeClear

clEventRetentionTimeClear

Synopsis:
Clears the retention event.

Header File:
clEventApi.h

Syntax:

```
ClRcT clEventRetentionTimeClear(
            CL_IN ClEventChannelHandleT channelHandle,
            CL_IN const ClEventIdT eventId );
```

Parameters:

channelHandle: (in) The handle of the event channel on which an event is published. `channelHandle` must be obtained earlier using the `clEventChannelOpen()` or `clEvtChannelOpenCallback()` functions.

eventId: (in) Identifier to the event.

Return values:

CL_OK: The function executed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

Description:
This function is used to clear the retention time of a published event, identified by `eventId`. It indicates that Event Service need not retain the event any longer for potential new subscribers. The event is no longer available for new subscribers when the value of the retention time is reset to zero,

Library File:
ClEventClient

Related function(s):
clEventPublish, clEvtEventDeliverCallbackT

3.4.9 clEventExtSubscribe

clEventExtSubscribe

Synopsis:
Subscribes to an event identified by event type (constant integer instead of filter).

Header File:
clEventExtApi.h

Syntax:

```
ClRcT clEventExtSubscribe(
            CL_IN ClEventChannelHandleT channelHandle,
            CL_IN ClUint32T eventType,
            CL_IN ClEventSubscriptionIdT subscriptionId,
            CL_IN void* pCookie );
```

Parameters:

channelHandle: (in) Handle of the event channel. The component subscribes to receive events on this event channel. This parameter must be obtained using the `clEventChannelOpen` or `clEvtChannelOpenCallback()` functions.

eventType: (in) Event type within event channel.

pSubscriptionId: (in) Identifies a specific subscription on this instance of the opened event channel (channelHandle), used as a parameter to `clEvtEventDeliverCallback()`.

pCookie: (in) Cookie, that is required to be provided by the component. It can be retrieved using the `clEvtEventUtilsCookieGet()` function.

Return values:

CL_OK: The function completed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_EVENT_ERR_NOT_OPENED_FOR_SUBSCRIPTION: The channel, identified by `channelHandle`, is not opened with the `CL_EVENT_CHANNEL_SUBSCRIBER` flag SET. The event channel is not opened to be accessed by subscribers.

Description:

This function serves as a wrapper around the `clEventSubscribe()` function and enables a component to subscribe for events on an event channel by registering a fixed length event type on that event channel instead of filters. Events are delivered using the `clEvtEventDeliverCallback()` callback function provided by the component when it calls the `clEventInitialize()` function.

The component must open the event channel, identified by channelHandle, with the `CL_EVENT_CHANNEL_SUBSCRIBER` flag set for successful execution of this function. The memory associated with the filters is not de-allocated by the `clEventExtSubscribe()` function. It is the responsibility of the invoking component to de-allocate the memory when the `clEventExtSubscribe()` function returns.

For a given subscription, the `pFilters` parameter cannot be modified. To change the filters parameter without losing events, a component must establish a new subscription (or

subscribe again) by assigning new values to the `pfilters` parameter. When a new subscription is established, the old subscription can be removed using the `clEventUnsubscribe()` function.

Library File:
ClEventClient

Related Function(s):
clEventSubscribe, clEventExtWithRbeSubscribe

3.4.10 clEventExtWithRbeSubscribe

clEventExtWithRbeSubscribe

Synopsis:
Subscribes to an event identified by the Rule Based Engine.

Header File:
clEventExtApi.h

Syntax:

```
ClRcT clEventExtWithRbeSubscribe(
            CL_IN const ClEventChannelHandleT channelHandle,
            CL_IN ClRuleExprT *pRbeExpr,
            CL_IN ClEventSubscriptionIdT subscriptionID,
            CL_IN void* pCookie);
```

Parameters:

channelHandle: (in) Handle of the event channel. The component subscribes to receive events on this event channel. This parameter must be obtained using `clEventChannelOpen()` or `clEvtChannelOpenCallback()` functions.

pRbeExpr: (in) Pointer to RBE expression allocated by the component. This defines filter patterns using the RBE expression. The component can de-allocate the memory for the filters when `clEventExtWithRbeSubscribe()` returns.

Identifies a specific subscription on this instance of the opened event channel (`channelHandle`), that is used as a parameter to `clEvtEventDeliverCallback()` function.

pCookie: (in) Cookie, provided to the function. This can be retrieved using the `clEvtEventUtilsCookiGet()` function.

Return values:

CL_OK: The function completed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

CL_EVENT_ERR_NOT_OPENED_FOR_SUBSCRIPTION: The channel, identified by `channelHandle`, is not opened with the `CL_EVENT_CHANNEL_SUBSCRIBER` flag set. The event channel is not opened to be accessed by subscribers.

Description:

This function is an extension to the `clEventSubscribe()` function. This function enables a component to subscribe for events on an event channel by registering RBE expression on that event channel.
Events are delivered using `clEvtEventDeliverCallback()` callback function, provided when the component calls the `clEventInitialize()` function.
The component must have opened the event channel, identified by `channelHandle`, with the `CL_EVENT_CHANNEL_SUBSCRIBER` flag set, for successful execution of the function.
The memory associated with the RBE is not de-allocated by the `clEventExtWithRbeSubscribe()` function. It is the responsibility of the invoking component to de-allocate the memory when the `clEventExtWithRbeSubscribe()`

function returns. For a given subscription, the filters parameter cannot be modified. To change the `pFilters` parameter without losing events, a component must establish a new subscription with the new filters parameter. After the new subscription is established, the old subscription can be removed using the `clEventUnsubscribe()` function.

Library File:
ClEventClient

Related Function(s):
clEventSubscribe,clEventExtSubscribe

3.4.11 clEventExtAttributesSet

clEventExtAttributesSet

Synopsis:
Sets (assigns values) the attributes of an event.

Header File:
clEventExtApi.h

Syntax:

```
ClRcT clEventExtAttributesSet(
            CL_IN ClEventHandleT eventHandle,
            CL_IN ClUint32T eventType,
            CL_IN ClEventPriorityT priority,
            CL_IN ClTimeT retentionTime,
            CL_IN const ClNameT *pPublisherName );
```

Parameters:

eventHandle: (in) Handle of the event whose attributes are to be set.

eventType: (in) Event type that needs to be published on a given channel.

priority: (in) Priority of the event.

retentionTime: (in) Duration for which the event is retained.

pPublisherName: (in) Pointer to the name of the publisher of the event.

Return values:

CL_OK: The function completed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

Description:
This function serves as a wrapper around `clEventAttributesSet()` taking a fixed length event type instead of a filter. It is used to set all the writeable event attributes such as fixed length event type, priority, `retentionTime`, and `pPublisherName`, in the header of the event, identified by the `eventHandle`. If `pPatternArray` or `pPublisherName` is NULL, the corresponding attributes are not changed.

Library File:
ClEventClient

Related Function(s):
clEventAttributesSet, clEventExtAttributesGet

3.4.12 clEventExtAttributesGet

clEventExtAttributesGet

Synopsis:

Returns the attributes of an event. It provides a component with the priority, retention time, publisher name, publish time, and the event identifier for an event.

Header File:

clEventExtApi.h

Syntax:

```
ClRcT clEventExtAttributesGet(
            CL_IN ClEventHandleT eventHandle,
            CL_OUT ClUint32T* pEventType,
            CL_OUT ClEventPriorityT *pPriority,
            CL_OUT ClTimeT *pRetentionTime,
            CL_OUT ClNameT *pPublisherName,
            CL_OUT ClTimeT *pPublishTime,
            CL_OUT ClEventIdT *pEventId );
```

Parameters:

eventHandle: (in) Handle of the event for which attributes are to be retrieved.

pEventType: (out) Pointer to type of an event.

pPriority: (out) Pointer to the priority of the event.

pRetentionTime: (out) Pointer to the duration for which the publisher will retain the event.

pPublisherName: (in) Pointer to the name of the publisher of the event.

pPublishTime: (out) Pointer to the time when the publisher published the event.

pEventId: (out) Pointer to the event identifier.

Return values:

CL_OK: The function completed successfully.

CL_EVENT_ERR_INIT_NOT_DONE: Event library is not initialized.

CL_EVENT_ERR_BAD_HANDLE: `eventHandle` is an invalid handle.

CL_EVENT_INTERNAL_ERROR: An unexpected problem occurred in the Event service.

CL_EVENT_ERR_INVALID_PARAM: An invalid parameter is passed to the function. A parameter is not set correctly.

Description:

This function servers as a wrapper around `clEventAttributesGet()` taking a fixed length event type instead of a filter. It retrieves the value of the attributes of the event, identified by `eventHandle`. For every `out` or `inout` parameters, if the invoking component provides a NULL reference, the Event Service does not return the `out` value. This function can be called on any event allocated by the `clEventAllocate()` function or received through the `clEvtEventDeliverCallback()` function. It can also be modified by the `clEventExtAttributesSet()` function.
If this function is invoked on a received event, the attributes `publish time` and `eventid` will have the values set by the Event Service at event publishing time.
Otherwise, the attributes will either have the initial values set by the Event Service when allocating the event, or the attributes set by a prior invocation of the `clEventExtAttributesSet()` function.

Library File:
ClEventClient

Related Function(s):
clEventAttributesSet, clEventExtAttributesGet

Chapter 4

Service Management Information Model

TBD

Chapter 5

Service Notifications

TBD

Chapter 6

Configuration

TBD

Chapter 7

Debug CLIs

TBD

Index