CMIDAPI.h File Reference

Typedefs

typedef enum **cmid_result_e**	**cmid_result_t**
typedef enum **cmid_url_type_e**	**cmid_url_type_t**

Enumerations

enum	**cmid_result_e** { **CMID_RES_SUCCESS** = 0 , **CMID_RES_GENERAL_ERROR** = -1 , **CMID_RES_NOT_INITED** = -2 , **CMID_RES_INVALID_ARG** = -3 , **CMID_RES_INSUFFICIENT_LEN** = -4 , **CMID_RES_AGENT_ERROR** = -5 , **CMID_RES_CLOUD_ERROR** = -6 , **CMID_RES_CLOUD_FAILURE** = -7 }
enum	**cmid_url_type_e** { **CMID_EVENT_URL** = 1 , **CMID_CHECKIN_URL** = 2 , **CMID_CATALOG_URL** = 3 }

Functions

CMID_CAPI **cmid_result_t**	**cmid_get_id** (IN OUT char *p_id, IN OUT int *p_buflen)
CMID_CAPI **cmid_result_t**	**cmid_get_token** (IN OUT char *p_token, IN OUT int *p_buflen)
CMID_CAPI **cmid_result_t**	**cmid_refresh_token** ()
CMID_CAPI **cmid_result_t**	**cmid_get_business_id** (IN OUT char *p_bid, IN OUT int *p_buflen)
CMID_CAPI **cmid_result_t**	**cmid_get_url** (IN **cmid_url_type_t** urlType, IN OUT char *p_url, IN OUT int *p_buflen)

Detailed Description

CMIDAPI has functions for use by clients to get the Business ID, CMID and its associated token. It is also possible to request for a refresh of the token associated with CMID. Additionaly, the event, catalog and checkin URLs can also be requested.

The CMID, Business ID and token are printable ASCII character sequences.

The contents of CMID, Business ID and token are opaque, and applications should not infer any meaning from their contents.

Memory Management:

- The caller of the API is responsible for allocating and freeing memory passed into the API.

Example usage:

```
#include "CMIDAPI.h"

void func_that_needs_cmid()
{
    int bufsz = 0;

    // Get required size
    cmid_result_t res = cmid_get_id(NULL, &bufsz);

    if (res == CMID_RES_INSUFFICIENT_LEN) {
        // Allocate memory of bufsz bytes
        char* mycmid = (char*) malloc(bufsz);
        // make sure mycmid is not NULL
        ...

        // Get CMID
        res = cmid_get_id(mycmid, &bufsz);
    }

    if (res != CMID_RES_SUCCESS) {
        // Handle failure
        return;
    }

    // Use mycmid
    // NOTE: bufsz includes 1 for terminating NUL
}
```

Typedef Documentation

◆ cmid_result_t

typedef enum **cmid_result_e** **cmid_result_t**

An enumeration to indicate the result of a CMID API call.

◆ cmid_url_type_t

typedef enum **cmid_url_type_e** **cmid_url_type_t**

An enumeration to indicate the URL types returned by cmid_get_url.

Enumeration Type Documentation

◆ cmid_result_e

enum **cmid_result_e**

An enumeration to indicate the result of a CMID API call.

Enumerator	
CMID_RES_SUCCESS	the call succeeded
CMID_RES_GENERAL_ERROR	the call failed due to an error other than these listed
CMID_RES_NOT_INITED	the API is not ready
CMID_RES_INVALID_ARG	an argument passed to the API is invalid
CMID_RES_INSUFFICIENT_LEN	the length of the memory block is not sufficient
CMID_RES_AGENT_ERROR	problem when communicating with the agent
CMID_RES_CLOUD_ERROR	problem in agent's communication with the cloud
CMID_RES_CLOUD_FAILURE	the cloud returned a failure response

◆ cmid_url_type_e

enum **cmid_url_type_e**

An enumeration to indicate the URL types returned by cmid_get_url.

Enumerator	
CMID_EVENT_URL	indicates the un-versioned event url
CMID_CHECKIN_URL	indicates the check-in url
CMID_CATALOG_URL	indicates the catalog url

Function Documentation

◆ cmid_get_business_id()

```
CMID_CAPI cmid_result_t cmid_get_business_id ( IN OUT char *  p_bid,
                                                IN OUT int *   p_buflen
                                              )
```

Get the Business ID.

Copies Business ID to the memory pointed to by `p_bid`. The terminating NUL character is also copied.

If `p_buflen` is NULL, then returns `CMID_RES_INVALID_ARG`. If `p_bid` is NULL, then updates `*p_buflen` with the size in bytes (including 1 for the terminating NUL) needed to store business ID. If both `p_bid` and `p_buflen` are not NULL, then `*p_buflen` should contain the size in bytes pointed to by `p_bid`. The business ID (including the terminating NUL) is copied to `p_bid` and `*p_buflen` is updated with the size of the business ID (including 1 for terminating NUL).

Parameters

`[in,out]`	**p_bid**	pointer to memory that can store the business ID
`[in,out]`	**p_buflen**	a non-NULL pointer to an integer

Returns

- CMID_RES_SUCCESS if the call is successful.
- CMID_RES_INVALID_ARG if `p_buflen` is NULL.
- CMID_RES_NOT_INITED if business id is not available.
- CMID_RES_INSUFFICIENT_LEN if `p_bid` is NULL, OR if `p_bid` is not NULL and `*p_buflen` does not have a value >= (size of business ID in bytes +1 for terminating NUL); `*p_buflen` is updated with the required size (including 1 for the terminating NUL).
- CMID_RES_GENERAL_ERROR if any other error occurs.

◆ cmid_get_id()

```
CMID_CAPI cmid_result_t cmid_get_id ( IN OUT char *  p_id,
                                      IN OUT int *   p_buflen
                                    )
```

Get the CMID.

Copies CMID to the memory pointed to by `p_id`. The terminating NUL character is also copied.

If `p_buflen` is NULL, then returns `CMID_RES_INVALID_ARG`. If `p_id` is NULL, then updates `*p_buflen` with the size in bytes (including 1 for the terminating NUL) needed to store CMID. If both `p_id` and `p_buflen` are not NULL, then `*p_buflen` should contain the size in bytes pointed to by `p_id`. The CMID (including the terminating NUL) is copied to `p_id` and `*p_buflen` is updated with the size of the CMID (including 1 for terminating NUL).

Parameters

`[in,out]`	**p_id**	pointer to memory that can store the CMID
`[in,out]`	**p_buflen**	a non-NULL pointer to an integer

Returns

- CMID_RES_SUCCESS if the call is successful.
- CMID_RES_INVALID_ARG if `p_buflen` is NULL.
- CMID_RES_NOT_INITED if CMID is not yet available.
- CMID_RES_INSUFFICIENT_LEN if `p_id` is NULL, OR if `p_id` is not NULL and `*p_buflen` does not have a value >= (size of CMID in bytes +1 for terminating NUL); `*p_buflen` is updated with the required size (including 1 for the terminating NUL).
- CMID_RES_GENERAL_ERROR if any other error occurs.

◆ cmid_get_token()

```
CMID_CAPI cmid_result_t cmid_get_token ( IN OUT char *  p_token,
                                         IN OUT int *   p_buflen
                                       )
```

Get the token that is associated with the CMID.

Copies the token associated with the CMID to the memory pointed to by `p_token`. The terminating NUL character is also copied.

If `p_buflen` is NULL, then returns `CMID_RES_INVALID_ARG`. If `p_token` is NULL, then updates `*p_buflen` with the size in bytes (including 1 for the terminating NUL) needed to store the token. If both `p_token` and `p_buflen` are not NULL, then `*p_buflen` should contain the size in bytes pointed to by `p_token`. The token (including the terminating NUL) is copied to `p_token` and `*p_buflen` is updated with the size of the token (including 1 for the terminating NUL).

Parameters

`[in,out]`	**p_token**	pointer to memory that can store the token
`[in,out]`	**p_buflen**	a non-NULL pointer to an integer

Returns

- CMID_RES_SUCCESS if the call is successful.
- CMID_RES_INVALID_ARG if `p_buflen` is NULL.
- CMID_RES_NOT_INITED if token is not yet available.
- CMID_RES_INSUFFICIENT_LEN if `p_token` is NULL, OR if `p_token` is not NULL and `*p_buflen` does not have a value >= (size of token in bytes +1 for terminating NUL); `*p_buflen` is updated with the required size (including 1 for the terminating NUL).
- CMID_RES_GENERAL_ERROR if any other error occurs.

◆ cmid_get_url()

```
CMID_CAPI cmid_result_t cmid_get_url ( IN cmid_url_type_t  urlType,
                                       IN OUT char *       p_url,
                                       IN OUT int *        p_buflen
                                     )
```

Get the event, catalog or checkin url based on the urlType param.

Copies the requested URL to the memory pointed to by `p_url`. The terminating NUL character is also copied.

If `p_buflen` is NULL, then returns `CMID_RES_INVALID_ARG`. If `p_url` is NULL, then updates `*p_buflen` with the size in bytes (including 1 for the terminating NUL) needed to store the url. If both `p_url` and `p_buflen` are not NULL, then `*p_buflen` should contain the size in bytes pointed to by `p_url`. The requested url (including the terminating NUL) is copied to `p_url` and `*p_buflen` is updated with the size of the url (including 1 for terminating NUL). If `urlType` is not supported, then returns CMID_RES_INVALID_ARG

Parameters

`[in]`	**urlType**	enum depicting the requested url type
`[in,out]`	**p_url**	pointer to memory that can store the requested url
`[in,out]`	**p_buflen**	a non-NULL pointer to an integer

Returns

- CMID_RES_SUCCESS if the call is successful.
- CMID_RES_INVALID_ARG if `p_buflen` is NULL, OR if `urlType` is invalid or not supported
- CMID_RES_NOT_INITED if the url is not available.
- CMID_RES_INSUFFICIENT_LEN if `p_url` is NULL, OR if `p_url` is not NULL and `*p_buflen` does not have a value >= (size of requested url in bytes +1 for terminating NUL); `*p_buflen` is updated with the required size (including 1 for the terminating NUL).
- CMID_RES_GENERAL_ERROR if any other error occurs.

◆ cmid_refresh_token()

CMID_CAPI **cmid_result_t** cmid_refresh_token ( )

Refresh the token that is associated with the CMID.

This call blocks until it gets a response.

Returns

- CMID_RES_SUCCESS if the call is successful.
- CMID_RES_AGENT_ERROR if there is a problem when communicating with the agent.
- CMID_RES_CLOUD_ERROR if there is a problem in agent's communication with the cloud.
- CMID_RES_CLOUD_FAILURE if the cloud returned a failure response.
- CMID_RES_GENERAL_ERROR if any other error occurs.

Generated by doxygen 1.9.1