

UCIDAPI.h File Reference

Typedefs

```
typedef enum ucid_result_e ucid_result_t
typedef enum ucid_url_type_e ucid_url_type_t
```

Enumerations

```
enum ucid_result_e {
    UCID_RES_SUCCESS = 0, UCID_RES_GENERAL_ERROR = -1,
    UCID_RES_NOT_INITED = -2, UCID_RES_INVALID_ARG = -3,
    UCID_RES_INSUFFICIENT_LEN = -4, UCID_RES_AGENT_ERROR = -5,
    UCID_RES_CLOUD_ERROR = -6, UCID_RES_CLOUD_FAILURE = -7
}
enum ucid_url_type_e { UCID_EVENT_URL = 1, UCID_CHECKIN_URL = 2,
    UCID_CATALOG_URL = 3 }
```

Functions

```
UCID_CAPI ucid_result_t ucid_get_id (IN OUT char *p_id, IN OUT int *p buflen)
UCID_CAPI ucid_result_t ucid_get_token (IN OUT char *p_token, IN OUT int *p buflen)
UCID_CAPI ucid_result_t ucid_refresh_token ()
UCID_CAPI ucid_result_t ucid_get_business_id (IN OUT char *p_bid, IN OUT int *p buflen)
UCID_CAPI ucid_result_t ucid_get_url (IN ucid_url_type_t urlType, IN OUT char *p_url, IN OUT int *p buflen)
```

Detailed Description

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

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

The contents of UCID, 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 "ucidapi.h"

void func_that_needs_ucid()
{
    int bufsz = 0;

    // Get required size
    ucid_result_t res = ucid_get_id(NULL, &bufsz);

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

        // Get UCID
        res = ucid_get_id(myucid, &bufsz);
    }

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

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

Copyright (c) 2020 Cisco Systems, Inc. All rights reserved.

Typedef Documentation

◆ ucid_result_t

```
typedef enum ucid_result_e ucid_result_t
```

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

◆ ucid_url_type_t

```
typedef enum ucid_url_type_e ucid_url_type_t
```

An enumeration to indicate the URL types returned by ucid_get_url.

Enumeration Type Documentation

◆ ucid_result_e

enum **ucid_result_e**

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

Enumerator

UCID_RES_SUCCESS	the call succeeded
UCID_RES_GENERAL_ERROR	the call failed due to an error other than these listed
UCID_RES_NOT_INITED	the API is not ready
UCID_RES_INVALID_ARG	an argument passed to the API is invalid
UCID_RES_INSUFFICIENT_LEN	the length of the memory block is not sufficient
UCID_RES_AGENT_ERROR	problem when communicating with the agent
UCID_RES_CLOUD_ERROR	problem in agent's communication with the cloud
UCID_RES_CLOUD_FAILURE	the cloud returned a failure response

◆ ucid_url_type_e

enum **ucid_url_type_e**

An enumeration to indicate the URL types returned by ucid_get_url.

Enumerator

UCID_EVENT_URL	indicates the un-versioned event url
UCID_CHECKIN_URL	indicates the check-in url
UCID_CATALOG_URL	indicates the catalog url

Function Documentation

◆ ucid_get_business_id()

```
UCID_CAPI ucid_result_t ucid_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 `UCID_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

- `UCID_RES_SUCCESS` if the call is successful.
- `UCID_RES_INVALID_ARG` if `p_buflen` is NULL.
- `UCID_RES_NOT_INITED` if business id is not available.
- `UCID_RES_INSUFFICIENT_LEN` if `p_bid` is NULL, OR if `p_bid` is not NULL and `*p_buflen` does not have a value \geq (size of business ID in bytes +1 for terminating NUL); `*p_buflen` is updated with the required size (including 1 for the terminating NUL).
- `UCID_RES_GENERAL_ERROR` if any other error occurs.

◆ ucid_get_id()

```
UCID_CAPI ucid_result_t ucid_get_id ( IN OUT char * p_id,  
                                     IN OUT int * p_buflen  
                                     )
```

Get the UCID.

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

If `p_buflen` is NULL, then returns `UCID_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 UCID. 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 UCID (including the terminating NUL) is copied to `p_id` and `*p_buflen` is updated with the size of the UCID (including 1 for terminating NUL).

Parameters

[in,out] `p_id` pointer to memory that can store the UCID

[in,out] `p_buflen` a non-NULL pointer to an integer

Returns

- `UCID_RES_SUCCESS` if the call is successful.
- `UCID_RES_INVALID_ARG` if `p_buflen` is NULL.
- `UCID_RES_NOT_INITED` if UCID is not yet available.
- `UCID_RES_INSUFFICIENT_LEN` if `p_id` is NULL, OR if `p_id` is not NULL and `*p_buflen` does not have a value \geq (size of UCID in bytes +1 for terminating NUL); `*p_buflen` is updated with the required size (including 1 for the terminating NUL).
- `UCID_RES_GENERAL_ERROR` if any other error occurs.

◆ `ucid_get_token()`

```
UCID_CAPI ucid_result_t ucid_get_token ( IN OUT char * p_token,
                                         IN OUT int *   p_buflen
                                       )
```

Get the token that is associated with the UCID.

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

If `p_buflen` is NULL, then returns `UCID_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

- `UCID_RES_SUCCESS` if the call is successful.
- `UCID_RES_INVALID_ARG` if `p_buflen` is NULL.
- `UCID_RES_NOT_INITED` if token is not yet available.
- `UCID_RES_INSUFFICIENT_LEN` if `p_token` is NULL, OR if `p_token` is not NULL and `*p_buflen` does not have a value \geq (size of token in bytes +1 for terminating NUL); `*p_buflen` is updated with the required size (including 1 for the terminating NUL).
- `UCID_RES_GENERAL_ERROR` if any other error occurs.

◆ `ucid_get_url()`

```
UCID_CAPI ucid_result_t ucid_get_url ( IN ucid_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 `UCID_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 `UCID_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

- `UCID_RES_SUCCESS` if the call is successful.
- `UCID_RES_INVALID_ARG` if `p_buflen` is NULL, OR if `urlType` is invalid or not supported
- `UCID_RES_NOT_INITED` if the url is not available.
- `UCID_RES_INSUFFICIENT_LEN` if `p_url` is NULL, OR if `p_url` is not NULL and `*p_buflen` does not have a value \geq (size of requested url in bytes +1 for terminating NUL); `*p_buflen` is updated with the required size (including 1 for the terminating NUL).
- `UCID_RES_GENERAL_ERROR` if any other error occurs.

◆ ucid_refresh_token()

UCID_CAPI **ucid_result_t** ucid_refresh_token()

Refresh the token that is associated with the UCID.

This call blocks until it gets a response.

Returns

- UCID_RES_SUCCESS if the call is successful.
- UCID_RES_AGENT_ERROR if there is a problem when communicating with the agent.
- UCID_RES_CLOUD_ERROR if there is a problem in agent's communication with the cloud.
- UCID_RES_CLOUD_FAILURE if the cloud returned a failure response.
- UCID_RES_GENERAL_ERROR if any other error occurs.