

UCIDAPI.h File Reference

TypeDefs

```
typedef enum ucid_result_e ucid_result_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
}
```

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 ()
```

Detailed Description

UCIDAPI has functions for use by clients to get the UCID and its associated token. It is also possible to request for a refresh of the token associated with UCID.

The UCID and token are printable ASCII character sequences.

The contents of UCID 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.

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 |

Function Documentation

- ◆ [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_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.