



## IOWA API Reference

IOTEROP

# Contents

<b>1</b>	<b>System Abstraction Layer</b>	<b>1</b>
1.1	Presentation . . . . .	1
1.2	Data types . . . . .	3
1.2.1	iowa_connection_type_t . . . . .	3
1.2.2	iowa_security_operation_t . . . . .	3
1.2.3	iowa_psk_data_t . . . . .	3
1.2.4	iowa_certificate_data_t . . . . .	4
1.2.5	iowa_rpk_data_t . . . . .	4
1.2.6	iowa_oscore_data_t . . . . .	5
1.2.7	iowa_security_data_t . . . . .	5
1.3	API . . . . .	7
1.3.1	iowa_system_malloc . . . . .	7
1.3.2	iowa_system_free . . . . .	8
1.3.3	iowa_system_gettime . . . . .	9
1.3.4	iowa_system_reboot . . . . .	10
1.3.5	iowa_system_trace . . . . .	11
1.3.6	iowa_system_connection_open . . . . .	12
1.3.7	iowa_system_connection_send . . . . .	13
1.3.8	iowa_system_connection_get_peer_identifier . . . . .	14
1.3.9	iowa_system_connection_recv . . . . .	15
1.3.10	iowa_system_connection_select . . . . .	16
1.3.11	iowa_system_connection_interrupt_select . . . . .	18
1.3.12	iowa_system_connection_close . . . . .	19
1.3.13	iowa_system_queue_create . . . . .	20
1.3.14	iowa_system_queue_delete . . . . .	21
1.3.15	iowa_system_queue_enqueue . . . . .	22
1.3.16	iowa_system_queue_dequeue . . . . .	23
1.3.17	iowa_system_queue_peek . . . . .	24
1.3.18	iowa_system_queue_remove . . . . .	25
1.3.19	iowa_system_queue_backup . . . . .	26
1.3.20	iowa_system_queue_restore . . . . .	27
1.3.21	iowa_system_store_context . . . . .	28
1.3.22	iowa_system_retrieve_context . . . . .	29
1.3.23	iowa_system_mutex_lock . . . . .	30
1.3.24	iowa_system_mutex_unlock . . . . .	31
1.3.25	iowa_system_random_vector_generator . . . . .	32
1.3.26	iowa_system_security_data . . . . .	33
<b>2</b>	<b>Common API Reference</b>	<b>35</b>
2.1	Presentation . . . . .	35
2.2	Data types . . . . .	35
2.2.1	iowa_status_t . . . . .	35
2.2.2	iowa_context_t . . . . .	36
2.2.3	iowa_dm_operation_t . . . . .	36
2.2.4	iowa_bootstrap_operation_t . . . . .	37
2.2.5	iowa_lwm2m_data_type_t . . . . .	38
2.2.6	iowa_lwm2m_data_t . . . . .	39
2.2.7	iowa_lwm2m_object_link_t . . . . .	40

2.2.8	iowa_content_format_t . . . . .	40
2.2.9	iowa_lwm2m_uri_t . . . . .	41
2.2.10	iowa_response_content_t . . . . .	43
2.3	Callbacks . . . . .	44
2.3.1	iowa_response_callback_t . . . . .	44
2.3.2	iowa_load_callback_t . . . . .	45
2.3.3	iowa_save_callback_t . . . . .	46
2.4	API . . . . .	47
2.4.1	iowa_init . . . . .	47
2.4.2	iowa_step . . . . .	48
2.4.3	iowa_flush_before_pause . . . . .	49
2.4.4	iowa_stop . . . . .	50
2.4.5	iowa_close . . . . .	51
2.4.6	iowa_save_context . . . . .	52
2.4.7	iowa_save_context_snapshot . . . . .	53
2.4.8	iowa_load_context . . . . .	54
2.4.9	iowa_backup_register_callback . . . . .	55
2.4.10	iowa_backup_deregister_callback . . . . .	56
2.4.11	iowa_connection_closed . . . . .	57
<b>3</b>	<b>Client Mode API Reference</b>	<b>58</b>
3.1	Client pseudo code . . . . .	58
3.2	Data types . . . . .	60
3.2.1	iowa_device_info_t . . . . .	60
3.2.2	iowa_event_type_t . . . . .	61
3.2.3	iowa_event_t . . . . .	61
3.2.4	iowa_device_time_info_t . . . . .	63
3.2.5	iowa_ipso_timed_value_t . . . . .	64
3.2.6	iowa_sensor_t . . . . .	64
3.2.7	iowa_lwm2m_resource_desc_t . . . . .	64
3.2.8	iowa_sensor_uri_t . . . . .	65
3.3	Callbacks . . . . .	66
3.3.1	iowa_event_callback_t . . . . .	66
3.3.2	iowa_client_time_update_callback_t . . . . .	66
3.3.3	iowa_client_factory_reset_callback_t . . . . .	66
3.3.4	iowa_RWE_callback_t . . . . .	66
3.3.5	iowa_CD_callback_t . . . . .	67
3.3.6	iowa_RI_callback_t . . . . .	67
3.4	API . . . . .	69
3.4.1	iowa_client_configure . . . . .	69
3.4.2	iowa_client_new_incoming_connection . . . . .	72
3.4.3	iowa_client_add_bootstrap_server . . . . .	73
3.4.4	iowa_client_remove_bootstrap_server . . . . .	74
3.4.5	iowa_client_set_bootstrap_server_hold_off . . . . .	75
3.4.6	iowa_client_get_bootstrap_server_coap_peer . . . . .	76
3.4.7	iowa_client_add_server . . . . .	77
3.4.8	iowa_client_remove_server . . . . .	79
3.4.9	iowa_client_set_server_msisdn . . . . .	80
3.4.10	iowa_client_set_server_registration_behaviour . . . . .	81
3.4.11	iowa_client_set_server_communication_attempts . . . . .	82
3.4.12	iowa_client_get_server_coap_peer . . . . .	84

3.4.13	iowa_client_set_notification_default_periods . . . . .	85
3.4.14	iowa_client_use_reliable_notifications . . . . .	86
3.4.15	iowa_client_object_set_mode . . . . .	87
3.4.16	iowa_client_device_update_battery . . . . .	88
3.4.17	iowa_client_add_device_power_source . . . . .	90
3.4.18	iowa_client_remove_device_power_source . . . . .	92
3.4.19	iowa_client_update_device_power_source . . . . .	93
3.4.20	iowa_client_set_device_error_code . . . . .	94
3.4.21	iowa_client_clear_device_error_code . . . . .	96
3.4.22	iowa_client_update_device_time_information . . . . .	97
3.4.23	iowa_client_add_custom_object . . . . .	98
3.4.24	iowa_client_remove_custom_object . . . . .	100
3.4.25	iowa_client_object_resource_changed . . . . .	101
3.4.26	iowa_client_object_instance_changed . . . . .	102
3.4.27	iowa_client_notification_lock . . . . .	103
3.4.28	iowa_client_send_heartbeat . . . . .	104
3.4.29	iowa_client_send_sensor_data . . . . .	105
3.4.30	iowa_client_send_data . . . . .	107
3.5	Accelerometer Object API . . . . .	109
3.5.1	iowa_client_add_accelerometer_object . . . . .	109
3.5.2	iowa_client_remove_accelerometer_object . . . . .	111
3.5.3	iowa_client_accelerometer_update_axis . . . . .	112
3.6	Access Control List Object API . . . . .	113
3.6.1	iowa_client_acl_rights_server_set . . . . .	113
3.6.2	iowa_client_acl_rights_server_clear . . . . .	115
3.6.3	iowa_client_acl_rights_object_clear . . . . .	116
3.7	APN Connection Profile Object API . . . . .	117
3.7.1	Data Structures and Constants . . . . .	117
3.7.2	Callbacks . . . . .	119
3.7.3	API . . . . .	120
3.8	AT Command Object API . . . . .	128
3.8.1	Callbacks . . . . .	128
3.8.2	API . . . . .	129
3.9	Bearer Selection Object API . . . . .	132
3.9.1	Data Structures and Constants . . . . .	132
3.9.2	Callbacks . . . . .	133
3.9.3	API . . . . .	134
3.10	Cellular Connectivity Object API . . . . .	138
3.10.1	Data Structures and Constants . . . . .	138
3.10.2	Callbacks . . . . .	139
3.10.3	API . . . . .	140
3.11	Connectivity Monitoring Object API . . . . .	144
3.11.1	Data Structures and Constants . . . . .	144
3.11.2	API . . . . .	146
3.12	Connectivity Statistics Object API . . . . .	149
3.12.1	iowa_client_add_connectivity_stats_object . . . . .	149
3.12.2	iowa_client_remove_connectivity_stats_object . . . . .	151
3.12.3	iowa_client_connectivity_stats_update_sms . . . . .	152
3.12.4	iowa_client_connectivity_stats_update_ip_data . . . . .	153
3.13	Digital Output Object API . . . . .	154
3.13.1	Callbacks . . . . .	154

3.13.2	API	155
3.14	Firmware Update Object API	158
3.14.1	Data Structures and Constants	158
3.14.2	Callbacks	159
3.14.3	API	163
3.15	GPS Object API	168
3.15.1	iowa_client_add_gps_object	168
3.15.2	iowa_client_remove_gps_object	169
3.15.3	iowa_client_gps_update_location	170
3.15.4	iowa_client_gps_update_location_full	171
3.16	Gyrometer Object API	173
3.16.1	iowa_client_add_gyrometer_object	173
3.16.2	iowa_client_remove_gyrometer_object	175
3.16.3	iowa_client_gyrometer_update_axis	176
3.17	IPSO Objects	177
3.17.1	iowa_client_IPSO_add_sensor	177
3.17.2	iowa_client_IPSO_update_value	180
3.17.3	iowa_client_IPSO_update_values	181
3.17.4	iowa_client_IPSO_remove_sensor	183
3.18	Ligth Control Object API	184
3.18.1	Callbacks	184
3.18.2	API	185
3.19	Location Object API	189
3.19.1	iowa_client_add_location_object	189
3.19.2	iowa_client_remove_location_object	190
3.19.3	iowa_client_location_update	191
3.19.4	iowa_client_location_update_full	192
3.20	Magnetometer Object API	194
3.20.1	iowa_client_add_magnetometer_object	194
3.20.2	iowa_client_remove_magnetometer_object	195
3.20.3	iowa_client_magnetometer_update_values	196
3.21	Software Component Object API	197
3.21.1	Data Structures and Constants	197
3.21.2	Callbacks	198
3.21.3	API	199
3.22	Software Management Object API	204
3.22.1	Data Structures and Constants	204
3.22.2	Callbacks	206
3.22.3	API	209
3.23	MQTT Object API	216
3.23.1	Data Structures and Constants	216
3.23.2	Callbacks	218
3.23.3	API	219
<b>4</b>	<b>Server Mode API Reference</b>	<b>229</b>
4.1	Server pseudo code	229
4.2	Data types	230
4.2.1	iowa_supported_format_t	230
4.2.2	iowa_lwm2m_protocol_version_t	230
4.2.3	iowa_client_t	231
4.3	Callbacks	232

4.3.1	iowa_result_callback_t . . . . .	232
4.3.2	iowa_monitor_callback_t . . . . .	232
4.3.3	iowa_resource_type_callback_t . . . . .	233
4.3.4	iowa_verify_client_callback_t . . . . .	233
4.4	API . . . . .	235
4.4.1	iowa_server_configure . . . . .	235
4.4.2	iowa_server_set_verify_client_callback . . . . .	236
4.4.3	iowa_server_new_incoming_connection . . . . .	237
4.4.4	iowa_server_configure_data_push . . . . .	238
4.4.5	iowa_server_read . . . . .	239
4.4.6	iowa_server_observe . . . . .	241
4.4.7	iowa_server_observe_cancel . . . . .	243
4.4.8	iowa_server_write . . . . .	244
4.4.9	iowa_server_write_attributes_string . . . . .	246
4.4.10	iowa_server_dm_exec . . . . .	248
4.4.11	iowa_server_dm_create . . . . .	250
4.4.12	iowa_server_dm_delete . . . . .	252
4.4.13	iowa_server_dm_discover . . . . .	253
4.4.14	iowa_server_set_response_format . . . . .	255
4.4.15	iowa_server_set_payload_format . . . . .	256
4.4.16	iowa_server_create_registration_update_trigger_message . . . . .	257
4.4.17	iowa_server_close_client_connection . . . . .	258
<b>5</b>	<b>Bootstrap Server Mode API Reference</b>	<b>259</b>
5.1	Bootstrap Server pseudo code . . . . .	259
5.2	Callbacks . . . . .	259
5.2.1	iowa_bootstrap_result_callback_t . . . . .	260
5.3	API . . . . .	261
5.3.1	iowa_bootstrap_server_configure . . . . .	261
5.3.2	iowa_bootstrap_server_new_incoming_connection . . . . .	262
5.3.3	iowa_bootstrap_server_read . . . . .	263
5.3.4	iowa_bootstrap_server_write . . . . .	265
5.3.5	iowa_bootstrap_server_delete . . . . .	267
5.3.6	iowa_bootstrap_server_discover . . . . .	268
5.3.7	iowa_bootstrap_server_finish . . . . .	269
5.3.8	iowa_bootstrap_server_add_server . . . . .	270
5.3.9	iowa_bootstrap_server_remove_server . . . . .	272
5.3.10	iowa_bootstrap_server_add_bootstrap_server . . . . .	274
5.3.11	iowa_bootstrap_server_remove_bootstrap_server . . . . .	276
<b>6</b>	<b>CoAP API Reference</b>	<b>278</b>
6.1	CoAP client pseudo code . . . . .	278
6.2	Data types . . . . .	279
6.2.1	iowa_coap_peer_t . . . . .	279
6.2.2	iowa_coap_peer_event_t . . . . .	279
6.2.3	iowa_coap_message_t . . . . .	279
6.2.4	iowa_coap_setting_id_t . . . . .	280
6.3	Callbacks . . . . .	281
6.3.1	iowa_coap_result_callback_t . . . . .	281
6.3.2	iowa_coap_peer_event_callback_t . . . . .	281
6.4	API . . . . .	282

6.4.1	iowa_coap_peer_new . . . . .	282
6.4.2	iowa_coap_peer_delete . . . . .	283
6.4.3	iowa_coap_peer_configuration_set . . . . .	284
6.4.4	iowa_coap_peer_configuration_get . . . . .	285
6.4.5	iowa_coap_peer_connect . . . . .	286
6.4.6	iowa_coap_peer_disconnect . . . . .	287
6.4.7	iowa_coap_peer_get . . . . .	288
6.4.8	iowa_coap_message_get_payload . . . . .	289
6.4.9	iowa_coap_message_get_block_info . . . . .	290
6.4.10	iowa_coap_block_request_next . . . . .	291
6.4.11	iowa_coap_block_request_block_number . . . . .	292
6.5	Helper Functions . . . . .	294
6.5.1	iowa_coap_uri_parse . . . . .	294

## **7 Utils API Reference 295**

7.1	Data types . . . . .	295
7.1.1	iowa_list_t . . . . .	295
7.2	Callbacks . . . . .	296
7.2.1	iowa_list_node_free_callback_t . . . . .	296
7.2.2	iowa_list_node_find_callback_t . . . . .	297
7.3	API . . . . .	298
7.3.1	iowa_utils_base64_get_encoded_size . . . . .	298
7.3.2	iowa_utils_base64_get_decoded_size . . . . .	299
7.3.3	iowa_utils_base64_encode . . . . .	300
7.3.4	iowa_utils_base64_decode . . . . .	301
7.3.5	iowa_utils_uri_to_sensor . . . . .	302
7.3.6	iowa_utils_sensor_to_uri . . . . .	303
7.3.7	iowa_utils_list_add . . . . .	304
7.3.8	iowa_utils_list_remove . . . . .	305
7.3.9	iowa_utils_list_free . . . . .	306
7.3.10	iowa_utils_list_find . . . . .	307
7.3.11	iowa_utils_list_find_and_remove . . . . .	308
7.4	Example: Linked List usage . . . . .	309

## **8 IOWA Components 310**

8.1	Overview . . . . .	310
8.2	Logger Component . . . . .	311
8.2.1	Presentation . . . . .	311
8.2.2	Functions . . . . .	312

## **9 Deprecated API Reference 316**

9.1	Deprecated Compilation Flags . . . . .	316
9.1.1	IOWA_SINGLE_CONNECTION_MODE . . . . .	316
9.1.2	LWM2M_SINGLE_SERVER_MODE . . . . .	316
9.1.3	LWM2M_OLD_CONTENT_FORMAT_SUPPORT . . . . .	316
9.1.4	IOWA_LORAWAN_MINIMAL_SUPPORT . . . . .	316
9.1.5	LWM2M_NOTIFICATION_QUEUE_SIZE . . . . .	316
9.1.6	LWM2M_STORAGE_QUEUE_PEEK_SUPPORT . . . . .	317
9.1.7	Lwm2m features removal . . . . .	317

## **A Appendix A i**

### **License**

Any user of the software is presumed to have read his license before using it and to have accepted its terms. By using this software, you acknowledge that you are fully aware that its use is strictly regulated by the license agreement to which it is subject by SAS IOTEROP.

The license associated with this version of the software is an evaluation license allowing only internal exploration and prototyping. Any commercial use of the software is strictly subject to the prior obtaining of a different and specific license (commercial license), the terms and conditions of which are defined by SAS IOTEROP. Any use of the software not expressly authorized by SAS IOTEROP may constitute a counterfeit punishable under French and international laws directly engaging the responsibility of its author (Article L335-3 of the French Intellectual Property Code - Berne Convention).



# 1 | System Abstraction Layer

The functions explained below are defined inside the file *include/iowa\_platform.h*.

## 1.1 Presentation

To port IOWA to your platform, you have to implement the following functions.

```
void * iowa_system_malloc(size_t size);

void iowa_system_free(void * pointer);

int32_t iowa_system_gettime(void);

int iowa_system_connection_send(void * connP,
                                uint8_t * buffer,
                                size_t length,
                                void * userData);

int iowa_system_connection_recv(void * connP,
                                uint8_t * buffer,
                                size_t length,
                                void * userData);

int iowa_system_connection_select(void ** connArray,
                                   size_t connCount,
                                   int32_t timeout,
                                   void * userData);

void iowa_system_connection_close(void * connP,
                                   void * userData);
```

If you are implementing a LwM2M Client, you also have to implement these functions:

```
void iowa_system_reboot(void *userData);

void * iowa_system_connection_open(iowa_connection_type_t type,
                                    char * hostname,
                                    char * port,
                                    void * userData);
```

These other functions are optional to implement. See Below.

```
void iowa_system_trace(const char * format,
                      va_list varArgs);

void * iowa_system_queue_create(void * userData);

void iowa_system_queue_delete(void * queueP,
                              void * userData);

int iowa_system_queue_enqueue(void * queueP,
                              uint8_t * buffer,
                              size_t length,
```

```
        void * userData);

size_t iowa_system_queue_dequeue(void * queueP,
                                uint8_t * buffer,
                                size_t length,
                                void * userData);

size_t iowa_system_queue_peek(void * queueP,
                              uint8_t * buffer,
                              size_t length,
                              void * userData);

void iowa_system_queue_remove(void * queueP,
                              void * userData);

size_t iowa_system_queue_backup(void * queueP,
                                uint8_t * buffer,
                                size_t length,
                                void * userData);

void * iowa_system_queue_restore(uint8_t * buffer,
                                 size_t length,
                                 void * userData);

size_t iowa_system_store_context(uint8_t * bufferP,
                                 size_t length,
                                 void * userData);

size_t iowa_system_retrieve_context(uint8_t ** bufferP,
                                    void * userData);

void iowa_system_connection_interrupt_select(void * userData);

void iowa_system_mutex_lock(void * userData);

void iowa_system_mutex_unlock(void * userData);

size_t iowa_system_connection_get_peer_identifier(void * connP,
                                                  uint8_t * addrP,
                                                  size_t length,
                                                  void * userData);

int iowa_system_random_vector_generator(uint8_t * randomBuffer,
                                       size_t size,
                                       void * userData);

iowa_status_t iowa_system_security_data(const uint8_t * peerIdentity,
                                       size_t peerIdentityLen,
                                       iowa_security_operation_t securityOp,
                                       iowa_security_data_t * securityDataP,
                                       void * userDataP);
```

## 1.2 Data types

### 1.2.1 iowa\_connection\_type\_t

```
typedef enum
{
    IOWA_CONN_UNDEFINED = 0,
    IOWA_CONN_DATAGRAM,
    IOWA_CONN_STREAM,
    IOWA_CONN_LORAWAN,
    IOWA_CONN_SMS
} iowa_connection_type_t;
```

This is an enumeration of the following values:

#### **IOWA\_CONN\_UNDEFINED**

Connection type is unknown.

#### **IOWA\_CONN\_DATAGRAM**

UDP connection.

#### **IOWA\_CONN\_STREAM**

TCP connection.

#### **IOWA\_CONN\_LORAWAN**

LoRaWAN transport.

#### **IOWA\_CONN\_SMS**

SMS transport.

### 1.2.2 iowa\_security\_operation\_t

```
typedef enum
{
    IOWA_SEC_READ,
    IOWA_SEC_FREE,
    IOWA_SEC_CREATE,
    IOWA_SEC_DELETE,
    IOWA_SEC_CHECK
} iowa_security_operation_t;
```

The `iowa_security_operation_t` enumeration is used to tell which security operation is requested when the function `iowa_system_security_data` is called.

#### **IOWA\_SEC\_READ**

Read security keys from the Application.

#### **IOWA\_SEC\_FREE**

Free the data allocated if any in the security structure. Always call after **IOWA\_SEC\_READ**.

#### **IOWA\_SEC\_CREATE**

Add security keys to the Application.

#### **IOWA\_SEC\_DELETE**

Remove security keys from the Application.

#### **IOWA\_SEC\_CHECK**

Check if the security keys are present on the Application.

### 1.2.3 iowa\_psk\_data\_t

```
typedef struct
```

```
{
    uint8_t *identity;
    size_t  identityLen;
    uint8_t *privateKey;
    size_t  privateKeyLen;
} iowa_psk_data_t;
```

The `iowa_psk_data_t` structure is used to store the identity / private key pair information.

#### **identity**

Identity associated with the private key.

#### **identityLen**

Length of the identity.

#### **privateKey**

Private key associated with the identity.

#### **privateKeyLen**

Length of the private key.

### 1.2.4 iowa\_certificate\_data\_t

```
typedef struct
{
    uint8_t *caCertificate;
    size_t  caCertificateLen;
    uint8_t *certificate;
    size_t  certificateLen;
    uint8_t *privateKey;
    size_t  privateKeyLen;
} iowa_certificate_data_t;
```

The `iowa_certificate_data_t` structure is used to store information related to a certificate.

#### **caCertificate**

Certificate authority used to generate the certificate. Must respect the DER (Distinguished Encoding Rules) format.

#### **caCertificateLen**

Length of the certificate authority.

#### **certificate**

Certificate. Must respect the DER (Distinguished Encoding Rules) format.

#### **certificateLen**

Length of the certificate.

#### **privateKey**

Private key used to generate the certificate. Must respect the DER (Distinguished Encoding Rules) format.

#### **privateKeyLen**

Length of the private key.

### 1.2.5 iowa\_rpk\_data\_t

```
typedef struct
{
    uint8_t *publicKeyX;
    size_t  publicKeyXLen;
    uint8_t *publicKeyY;
    size_t  publicKeyYLen;
    uint8_t *privateKey;
```

```
size_t   privateKeyLen;
} iowa_rpk_data_t;
```

The `iowa_rpk_data_t` structure is used to store information related to raw public key information. Supported public keys must use the fixed curve *secp256r1*.

**publicKeyX**

Public key X coordinate.

**publicKeyXLen**

Length of the public key X coordinate.

**publicKeyY**

Public key Y coordinate.

**publicKeyYLen**

Length of the public key Y coordinate.

**privateKey**

Private key used to generate the public key.

**privateKeyLen**

Length of the private key.

### 1.2.6 iowa\_oscore\_data\_t

```
typedef struct
{
    uint8_t *senderId;
    size_t   senderIdLen;
    uint8_t *recipientId;
    size_t   recipientIdLen;
    uint8_t *masterSecret;
    size_t   masterSecretLen;
} iowa_oscore_data_t;
```

The `iowa_oscore_data_t` structure is used to store information related to OSCORE key information.

**senderId**

ID to use to protect sent CoAP messages.

**senderIdLen**

Length of the Sender ID.

**recipientId**

ID to use to verify received CoAP messages..

**recipientIdLen**

Length of the Recipient ID.

**masterSecret**

Private key associated to the Sender ID and Recipient ID.

**masterSecretLen**

Length of the Master Secret.

### 1.2.7 iowa\_security\_data\_t

```
typedef struct
{
    iowa_security_mode_t securityMode;
    union
    {
```

```
iowa_psk_data_t      pskData;  
iowa_certificate_data_t certData;  
iowa_rpk_data_t      rpkData;  
iowa_oscore_data_t   oscoreData;  
} protocol;  
} iowa_security_data_t;
```

The `iowa_security_data_t` structure is used to create, delete, read and free security data.

**securityMode**

Security mode to determine the data in the union structure.

**protocol.pskData**

Pre-shared key data.

**protocol.certData**

Certificate data.

**protocol.rpkData**

Raw public key data.

**protocol.oscoreData**

Object Security for CORE key data.

## 1.3 API

### 1.3.1 iowa\_system\_malloc

#### Prototype

```
void * iowa_system_malloc(size_t size);
```

#### Description

`iowa_system_malloc()` could map directly to C standard library `malloc()`. It allocates a memory block.

#### Arguments

##### *size*

The size in bytes of the requested memory block.

#### Return Value

A pointer to the allocated memory or NULL in case of error.

#### Header File

`iowa_platform.h`

#### Note

This function is required by IOWA.

### 1.3.2 iowa\_system\_free

#### Prototype

```
void iowa_system_free(void * pointer);
```

#### Description

`iowa_system_free()` could map directly to C standard library `free()`. It releases a memory block previously allocated by `iowa_system_malloc()`.

#### Arguments

##### *pointer*

A pointer to the memory block to release.

#### Return Value

None.

#### Header File

`iowa_platform.h`

#### Notes

This function is required by IOWA.

From C standard, no action should occur if *pointer* argument is a null pointer. This is how the function `free()` from C standard library behaves. However some compilers or implementations do not respect this standard. So it can be necessary when implementing `iowa_system_free()` to check if the argument *pointer* is nil.



### 1.3.3 iowa\_system\_gettime

#### Prototype

```
int32_t iowa_system_gettime(void);
```

#### Description

`iowa_system_gettime()` is used by IOWA to determine the time elapsed.

#### Return Value

The number of seconds elapsed since a point of origin or a negative number in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required by IOWA.

If you are using the GPS or the Location object, this function will be used to timestamp the measure. In this case, the point of origin must be Epoch.

Else, the point of origin (Epoch, system boot, etc...) does not matter as this function is used only to determine the elapsed time between consecutive calls.

There is no safeguard if `iowa_system_gettime()` returns a value inferior to the one returned in a previous call.

### 1.3.4 iowa\_system\_reboot

#### Prototype

```
void iowa_system_reboot(void *userData);
```

#### Description

`iowa_system_reboot()` starts a system reboot.

#### Arguments

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

None.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only for LwM2M Clients.

This feature is required by the Lightweight M2M protocol. However, a LwM2M device can be functional without it and this function can be a stub.

### 1.3.5 iowa\_system\_trace

#### Prototype

```
void iowa_system_trace(const char * format,  
                      va_list varArgs);
```

#### Description

`iowa_system_trace()` outputs the logs when the stack is built with **IOWA\_WITH\_LOGS**.

It can be mapped directly to `vprintf()`.

#### Return Value

None.

#### Header File

`iowa_platform.h`

#### Notes

This function is required by IOWA only when logs are activated. See `[IOWA_LOG_LEVEL and IOWA_LOG_PART][IOWA_LOG_LEVEL and IOWA_LOG_PART]`.

IOWA takes care of freeing `varArgs` after this call returns.

To print a single line on the output, this function can be called multiple times. This has an impact when the flag **IOWA\_THREAD\_SUPPORT** is enabled. Several calls of `iowa_system_trace()` can occur from different threads at the same time. And thus, the output can be ruined. The implementation of this function must be thread safe.

### 1.3.6 iowa\_system\_connection\_open

#### Prototype

```
void * iowa_system_connection_open(iowa_connection_type_t type,  
                                   char * hostname,  
                                   char * port,  
                                   void * userData);
```

#### Description

`iowa_system_connection_open()` opens a connection to a host.

#### Arguments

**type**

The type of connection to open.

**hostname**

The hostname of the peer to connect to.

**port**

The port to connect to. It may be NULL depending on the provided server URL.

**userData**

The argument passed to `iowa_init()`.

#### Return Value

A pointer to an user-defined type or NULL in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only for LwM2M Clients.

When IOWA is used as a LwM2M Client, it calls this function to connect to the LwM2M Servers. See the `iowa_client_add_server()` API.

### 1.3.7 iowa\_system\_connection\_send

#### Prototype

```
int iowa_system_connection_send(void * connP,  
                                uint8_t * buffer,  
                                size_t length,  
                                void * userData);
```

#### Description

`iowa_system_connection_send()` sends a buffer on a connection.

#### Arguments

##### ***connP***

The connection as returned by `iowa_system_connection_open()`.

##### ***buffer***

The data to send.

##### ***length***

The length of the data in bytes.

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

The number of bytes sent or a negative number in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required by IOWA.

On packet switched networks (eg. UDP), if *length* is bigger than the MTU, it is advised to not try to send the buffer and return the MTU.

### 1.3.8 iowa\_system\_connection\_get\_peer\_identifier

#### Prototype

```
size_t iowa_system_connection_get_peer_identifier(void * connP,
                                                uint8_t * addrP,
                                                size_t length,
                                                void * userData);
```

#### Description

`iowa_system_connection_get_peer_identifier()` returns an unique identifier for the peer of a connection (e.g. IP address, LoRaWAN DevEUI, SMS MSISDN).

#### Arguments

##### ***connP***

The connection as returned by `iowa_system_connection_open()`.

##### ***addrP***

A pre-allocated buffer to store the identifier.

##### ***length***

The length of *addrP* in bytes.

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

The number of bytes of the identifier or 0 in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with `[IOWA_SECURITY_LAYER][IOWA_SECURITY_LAYER]` different from **`IOWA_SECURITY_LAYER_NONE`**, or with **`LWM2M_SERVER_MODE`** or **`LWM2M_BOOTSTRAP_SERVER_MODE`** flags.

This is used when the endpoint name has not been found in the registration payload.

### 1.3.9 iowa\_system\_connection\_recv

#### Prototype

```
int iowa_system_connection_recv(void * connP,  
                                uint8_t * buffer,  
                                size_t length,  
                                void * userData);
```

#### Description

`iowa_system_connection_recv()` reads data from a connection in a non-blocking way.

#### Arguments

##### ***connP***

The connection as returned by `iowa_system_connection_open()`.

##### ***buffer***

A buffer to store the received data.

##### ***length***

The length of the buffer in bytes.

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

The number of received bytes or a negative number in case of error.

#### Header File

`iowa_platform.h`

#### Note

This function is required by IOWA.

### 1.3.10 iowa\_system\_connection\_select

#### Prototype

```
int iowa_system_connection_select(void ** connArray,
                                size_t connCount,
                                int32_t timeout,
                                void * userData);
```

#### Description

`iowa_system_connection_select()` monitors a list of connections for incoming data during the specified time.

#### Arguments

##### **connArray**

An array of connections as returned by `iowa_system_connection_open()`.

##### **connCount**

The number of elements of `connArray`. This may be zero.

##### **timeout**

The time to wait for data in seconds. This may be zero.

##### **userData**

The argument passed to `iowa_init()`.

##### **#### Return Value {-}**

either:

- a positive number if data are available.
- zero if the time elapsed.
- a negative number in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required by IOWA.

If data are available on one or more connections, `iowa_system_connection_select()` must modify the `connArray` elements :

- If data are available on a connection the matching element in `connArray` is left untouched.
- If no data are available, the matching element is set to NULL. If the timeout is reached (or in case of error), there is no need to modify the `connArray` elements.

When the application needs to be very responsive, this function is a good place to monitor application specific events without using a very short timeout in `iowa_step()`. For instance, the sample server waits for keyboard events here.

It is possible that IOWA calls `iowa_system_connection_select()` with a timeout of zero. In this case, `iowa_system_connection_select()` must not return an error, and should check if some data are already available on one of the connections.

It is also possible that IOWA calls `iowa_system_connection_select()` with no connections. In this case, `iowa_system_connection_select()` must not return an error, and should return after the timeout, which may also



be zero, expires. This can occur when opening a connection to a LwM2M Server failed and IOWA is configured to wait a specific time before retrying to connect to the LwM2M Server.

Evaluation

### 1.3.11 iowa\_system\_connection\_interrupt\_select

#### Prototype

```
void iowa_system_connection_interrupt_select(void * userData);
```

#### Description

A call to `iowa_system_connection_interrupt_select()` makes `iowa_system_connection_select()` return immediately if it is currently running.

#### Arguments

##### *userData*

The argument passed to `iowa_init()`.

#### Return Value

None.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **IOWA\_MULTITHREAD\_SUPPORT** flag.

The value returned by `iowa_system_connection_select()` is zero. Calling `iowa_system_connection_interrupt_select()` is considered as a preemptive timeout for `iowa_system_connection_select()`.

### 1.3.12 iowa\_system\_connection\_close

#### Prototype

```
void iowa_system_connection_close(void * connP,  
                                  void * userData);
```

#### Description

`iowa_system_connection_close()` closes a connection.

#### Arguments

##### ***connP***

The connection as returned by `iowa_system_connection_open()`.

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

None.

#### Header File

`iowa_platform.h`

#### Note

This function is required by IOWA.

### 1.3.13 iowa\_system\_queue\_create

#### Prototype

```
void * iowa_system_queue_create(void * userData);
```

#### Description

`iowa_system_queue_create()` creates a storage queue to offload data from the memory.

#### Arguments

##### *userData*

The argument passed to `iowa_init()`.

#### Return Value

A pointer to a user-defined type or NULL in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **LWM2M\_STORAGE\_QUEUE\_SUPPORT** or **LWM2M\_STORAGE\_QUEUE\_PEEK\_SUPPORT** flags.

The storage queue must store data as separate entities and not as a stream. Ideally, this is a FIFO.

### 1.3.14 iowa\_system\_queue\_delete

#### Prototype

```
void iowa_system_queue_delete(void * queueP,  
                             void * userData);
```

#### Description

`iowa_system_queue_delete()` closes a storage queue.

#### Arguments

##### ***queueP***

A storage queue as returned by `iowa_system_queue_create()`.

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

None.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **LWM2M\_STORAGE\_QUEUE\_SUPPORT** or **LWM2M\_STORAGE\_QUEUE\_PEEK\_SUPPORT** flags.

### 1.3.15 iowa\_system\_queue\_enqueue

#### Prototype

```
int iowa_system_queue_enqueue(void * queueP,
                             uint8_t * buffer,
                             size_t length,
                             void * userData);
```

#### Description

`iowa_system_queue_enqueue()` stores data in a storage queue.

#### Arguments

##### **queueP**

A storage queue as returned by `iowa_system_queue_create()`.

##### **buffer**

The data to store.

##### **length**

The length of the data in bytes.

##### **userData**

The argument passed to `iowa_init()`.

#### Return Value

The number of stored bytes or a negative number in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **LWM2M\_STORAGE\_QUEUE\_SUPPORT** or **LWM2M\_STORAGE\_QUEUE\_PEEK\_SUPPORT** flags.

The data must be stored as a single entity.

### 1.3.16 iowa\_system\_queue\_dequeue

#### Prototype

```
size_t iowa_system_queue_dequeue(void * queueP,  
                                uint8_t * buffer,  
                                size_t length,  
                                void * userData);
```

#### Description

`iowa_system_queue_dequeue()` retrieves an entity from a storage queue or, the provided buffer is too small, it returns the size of the next entity to retrieve.

#### Arguments

***queueP***

A storage queue as returned by `iowa_system_queue_create()`.

***buffer***

A buffer to store the retrieved data. This can be nil.

***length***

The length of the buffer in bytes. This can be zero.

***userData***

The argument passed to `iowa_init()`.

#### Return Value

The size in bytes of retrieved entity or zero if the queue is empty or in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **LWM2M\_STORAGE\_QUEUE\_SUPPORT** flag.

If the provided buffer is nil or too small to contain the entity to retrieve, the entity is not removed from the queue.

### 1.3.17 iowa\_system\_queue\_peek

#### Prototype

```
size_t iowa_system_queue_peek(void * queueP,  
                             uint8_t * buffer,  
                             size_t length,  
                             void * userData);
```

#### Description

`iowa_system_queue_peek()` peeks a entity from a storage queue or, if the provided buffer is too small, it returns the size of the next entity to peek.

#### Arguments

***queueP***

A storage queue as returned by `iowa_system_queue_create()`.

***buffer***

A buffer to store the retrieved data. This can be nil.

***length***

The length of the buffer in bytes. This can be zero.

***userData***

The argument passed to `iowa_init()`.

#### Return Value

The size in bytes of peeked entity or zero if the queue is empty or in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the `LWM2M_STORAGE_QUEUE_PEEK_SUPPORT` flag.



### 1.3.18 iowa\_system\_queue\_remove

#### Prototype

```
void iowa_system_queue_remove(void * queueP,  
                             void * userData);
```

#### Description

`iowa_system_queue_remove()` removes the first entity of a storage queue.

#### Arguments

##### ***queueP***

A storage queue as returned by `iowa_system_queue_create()`.

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

None.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **LWM2M\_STORAGE\_QUEUE\_PEEK\_SUPPORT** flag.

### 1.3.19 iowa\_system\_queue\_backup

#### Prototype

```
size_t iowa_system_queue_backup(void *queueP,  
                                uint8_t *buffer,  
                                size_t length,  
                                void *userData);
```

#### Description

`iowa_system_queue_backup()` returns a blob of data necessary to recreate a storage queue, or if the provided buffer is too small, the size of this blob of data.

#### Arguments

***queueP***

A storage queue as returned by `iowa_system_queue_create()`.

***buffer***

A buffer to store the blob of data. This can be nil.

***length***

The length of *buffer* in bytes. This can be zero.

***userData***

The argument passed to `iowa_init()`.

#### Return Value

The size in bytes of the blob of data to save or zero in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **IOWA\_STORAGE\_CONTEXT\_SUPPORT** flag and with the **LWM2M\_STORAGE\_QUEUE\_SUPPORT** or **LWM2M\_STORAGE\_QUEUE\_PEEK\_SUPPORT** flags.

### 1.3.20 iowa\_system\_queue\_restore

#### Prototype

```
void *iowa_system_queue_restore(uint8_t *buffer,  
                                size_t length,  
                                void *userData);
```

#### Description

`iowa_system_queue_restore()` recreates a storage queue from a blob of data.

#### Arguments

##### **buffer**

A buffer containing the blob of data returned by `iowa_system_queue_backup()`.

##### **length**

The length of *buffer* in bytes.

##### **userData**

The argument passed to `iowa_init()`.

#### Return Value

A pointer to an user-defined type or NULL in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **IOWA\_STORAGE\_CONTEXT\_SUPPORT** flag and with the **LWM2M\_STORAGE\_QUEUE\_SUPPORT** or **LWM2M\_STORAGE\_QUEUE\_PEEK\_SUPPORT** flags.

### 1.3.21 iowa\_system\_store\_context

#### Prototype

```
size_t iowa_system_store_context(uint8_t *bufferP,  
                                size_t length,  
                                void *userData);
```

#### Description

`iowa_system_store_context()` stores the IOWA context.

#### Arguments

***bufferP***

The destination buffer.

***length***

Length of the buffer.

***userData***

The argument passed to `iowa_init()`.

#### Return Value

The number of stored bytes or a zero in case of error.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **IOWA\_STORAGE\_CONTEXT\_SUPPORT** flag.

### 1.3.22 iowa\_system\_retrieve\_context

#### Prototype

```
size_t iowa_system_retrieve_context(uint8_t **bufferP,  
                                   void *userData);
```

#### Description

`iowa_system_retrieve_context()` retrieves an IOWA context.

#### Arguments

##### ***bufferP***

The buffer containing the retrieved data.

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

The size in bytes of retrieved data or zero if there is nothing or in case of error.

#### Header File

`iowa_platform.h`

#### Notes

- This function is required only if IOWA is built with the **IOWA\_STORAGE\_CONTEXT\_SUPPORT** flag.
- *bufferP* must be allocated by the function. The buffer will next be freed by IOWA internally.

### 1.3.23 iowa\_system\_mutex\_lock

#### Prototype

```
void iowa_system_mutex_lock(void * userData);
```

#### Description

`iowa_system_mutex_lock()` locks a mutex for the current thread.

#### Arguments

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

None.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **IOWA\_MULTITHREAD\_SUPPORT** flag.

IOWA uses only one mutex.

### 1.3.24 iowa\_system\_mutex\_unlock

#### Prototype

```
void iowa_system_mutex_unlock(void * userData);
```

#### Description

`iowa_system_mutex_unlock()` releases a mutex.

#### Arguments

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

None.

#### Header File

`iowa_platform.h`

#### Notes

This function is required only if IOWA is built with the **IOWA\_MULTITHREAD\_SUPPORT** flag.

IOWA uses only one mutex.

### 1.3.25 iowa\_system\_random\_vector\_generator

#### Prototype

```
int iowa_system_random_vector_generator(uint8_t *randomBuffer,
                                       size_t size,
                                       void *userData);
```

#### Description

`iowa_system_random_vector_generator()` stores random values to a preallocated vector.

#### Arguments

##### ***randomBuffer***

The generated random vector.

##### ***size***

The size of the vector.

##### ***userData***

The argument passed to `iowa_init()`.

#### Return Value

A code indicating if the vector has been generated:

- 0 if the vector has been generated successfully.
- Another value if an error occurred.

#### Header File

`iowa_platform.h`

#### Notes

This function is only required if IOWA is built with **IOWA\_SECURITY\_LAYER** different from **IOWA\_SECURITY\_LAYER\_NONE**.



### 1.3.26 iowa\_system\_security\_data

#### Prototype

```
iowa_status_t iowa_system_security_data(const uint8_t *peerIdentity,
                                         size_t peerIdentityLen,
                                         iowa_security_operation_t securityOp,
                                         iowa_security_data_t *securityDataP,
                                         void *userDataP);
```

#### Description

`iowa_system_security_data()` is a function used by the security layer to CREATE, DELETE or READ the security data.

#### Arguments

##### **peerIdentity**

The identity associating to the peer (can be an URI on client side, or the PSK identity on server side, etc).

##### **peerIdentitySize**

Size of the identity.

##### **securityOp**

The security operation.

##### **securityDataP**

The security data.

##### **userDataP**

The argument passed to `iowa_init()`.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

the security data based on *peerIdentity* has not been found (only when the security operation is not `IOWA_SEC_CREATE`).

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

the security data based on *peerIdentity* already exists (only when the security operation is `IOWA_SEC_CREATE`).

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`iowa_platform.h`

#### Notes

This function is only required if IOWA is built with `IOWA_SECURITY_LAYER` different from `IOWA_SECURITY_LAYER_NONE`.

The `IOWA_SEC_DELETE` operation is only used if a `IOWA_SEC_CREATE` operation happened. The `IOWA_SEC_CREATE` operation is only used on bootstrapping.

Regarding the memory allocation:

- On a `IOWA_SEC_CREATE` operation, the data is allocated by the stack. So the deallocation MUST not be done by the application.
- On a `IOWA_SEC_READ` operation, the data if needed is allocated by the application. The stack does not deallocate it. That's why after a `IOWA_SEC_READ` operation, the function `iowa_system_security_data` is called with `IOWA_SEC_FREE` to allow the application to deallocate the memory if needed.

*peerIdentity* is always a character string encoded in UTF-8 without the Null-terminated string:

- On a LwM2M Client, this is the LwM2M Server URI or the Bootstrap Server URI.
- On Server, this is Client Hint Identity.

Evaluation

## 2 | Common API Reference

The functions explained below are defined inside the file *include/iowa.h*.

### 2.1 Presentation

Whatever the role of your application (Client or Server), it builds on the following skeleton:

```
#include "iowa.h"

int main(int argc,
         char *argv[])
{
    iowa_context_t iowaH;
    iowa_status_t result;

    /*****
     * Initialization
     */

    iowaH = iowa_init(NULL);

    /*****
     * "Main loop"
     */

    do
    {
        result = iowa_step(iowaH, 5);
    } while (result == IOWA_COAP_NO_ERROR)

    iowa_close(iowaH);

    return 0;
}
```

### 2.2 Data types

#### 2.2.1 iowa\_status\_t

This is the return type of most of the IOWA APIs. This is an enumeration matching the CoAP status codes, similar to the HTTP status codes.

```
typedef uint8_t iowa_status_t;

#define IOWA_COAP_NO_ERROR          0x00
#define IOWA_COAP_201_CREATED      0x41
#define IOWA_COAP_202_DELETED      0x42
#define IOWA_COAP_203_VALID        0x43
#define IOWA_COAP_204_CHANGED      0x44
#define IOWA_COAP_205_CONTENT      0x45
#define IOWA_COAP_231_CONTINUE     0x5F
```

```
#define IOWA_COAP_400_BAD_REQUEST          0x80
#define IOWA_COAP_401_UNAUTHORIZED        0x81
#define IOWA_COAP_402_BAD_OPTION          0x82
#define IOWA_COAP_403_FORBIDDEN           0x83
#define IOWA_COAP_404_NOT_FOUND            0x84
#define IOWA_COAP_405_METHOD_NOT_ALLOWED  0x85
#define IOWA_COAP_406_NOT_ACCEPTABLE      0x86
#define IOWA_COAP_408_REQUEST_ENTITY_INCOMPLETE 0x88
#define IOWA_COAP_409_CONFLICT            0x89
#define IOWA_COAP_412_PRECONDITION_FAILED  0x8C
#define IOWA_COAP_413_REQUEST_ENTITY_TOO_LARGE 0x8D
#define IOWA_COAP_415_UNSUPPORTED_CONTENT_FORMAT 0x8F
#define IOWA_COAP_422_UNPROCESSABLE_ENTITY 0x96
#define IOWA_COAP_429_TOO_MANY_REQUESTS    0x9D
#define IOWA_COAP_500_INTERNAL_SERVER_ERROR 0xA0
#define IOWA_COAP_501_NOT_IMPLEMENTED      0xA1
#define IOWA_COAP_502_BAD_GATEWAY          0xA2
#define IOWA_COAP_503_SERVICE_UNAVAILABLE  0xA3
#define IOWA_COAP_504_GATEWAY_TIMEOUT      0xA4
#define IOWA_COAP_505_PROXYING_NOT_SUPPORTED 0xA5
```

### 2.2.2 iowa\_context\_t

This opaque type is used to store the context of the IOWA stack engine. It is created by calling `iowa_init()` and destroyed by calling `iowa_close()`.

Multiple `iowa_context_t` can be created within the same process.

### 2.2.3 iowa\_dm\_operation\_t

```
typedef uint8_t iowa_dm_operation_t;

#define IOWA_DM_UNDEFINED          0
#define IOWA_DM_READ              1
#define IOWA_DM_FREE              2
#define IOWA_DM_WRITE             3
#define IOWA_DM_EXECUTE           4
#define IOWA_DM_CREATE            5
#define IOWA_DM_DELETE            6
#define IOWA_DM_DISCOVER          7
#define IOWA_DM_WRITE_ATTRIBUTES  8
#define IOWA_DM_NOTIFY            9
#define IOWA_DM_CANCEL           10
#define IOWA_DM_DATA_PUSH        11
#define IOWA_DM_READ_REQUEST     12
```

This is an enumeration of the following values:

#### **IOWA\_DM\_UNDEFINED**

No specific LwM2M operation, this should never be used and only serves as a non default operation.

#### **IOWA\_DM\_READ**

LwM2M Read operation is used to access the value of an Object, Object Instances, and Resources.

#### **IOWA\_DM\_FREE**

Free operation is used to clean the allocated memory on `IOWA_DM_READ` operation.

### **IOWA\_DM\_WRITE**

LwM2M Write operation is used to change the value of an Object, Object Instances, and Resources.

### **IOWA\_DM\_EXECUTE**

LwM2M Execute operation is used to initiate some action, and can only be performed on individual Resources.

### **IOWA\_DM\_CREATE**

LwM2M Create operation is used to create Object Instance(s).

### **IOWA\_DM\_DELETE**

LwM2M Delete operation is used to delete an Object Instance.

### **IOWA\_DM\_DISCOVER**

LwM2M Discover operation is used to discover LwM2M Attributes attached to an Object, Object Instances, and Resources.

### **IOWA\_DM\_WRITE\_ATTRIBUTES**

LwM2M Write-Attributes operation is used to change the LwM2M Attributes of an Object, Object Instances, and Resources.

### **IOWA\_DM\_NOTIFY**

LwM2M Notify operation is used to notify the change of a value during a valid Observation on an Object Instance or Resource.

### **IOWA\_DM\_CANCEL**

LwM2M Cancel operation is used to end an Observation.

### **IOWA\_DM\_DATA\_PUSH**

LwM2M Data push operation is used when LwM2M Server receives the value of an Object, Object Instances, and Resources without requested it.

### **IOWA\_DM\_READ\_REQUEST**

On the Client side, inform a custom Object that a LwM2M Server will perform a READ operation. See [iowa\\_lwm2m\\_resource\\_desc\\_t](#).

#### **2.2.3.1 Notes**

This enumeration is used on both Server and Client side. But not all the values are used depending of the side.

Server callbacks can be called with:

- IOWA\_DM\_READ
- IOWA\_DM\_WRITE
- IOWA\_DM\_EXECUTE
- IOWA\_DM\_CREATE
- IOWA\_DM\_DELETE
- IOWA\_DM\_DISCOVER
- IOWA\_DM\_WRITE\_ATTRIBUTES
- IOWA\_DM\_NOTIFY
- IOWA\_DM\_DATA\_PUSH

Client callbacks can be called with:

- IOWA\_DM\_READ
- IOWA\_DM\_FREE
- IOWA\_DM\_WRITE
- IOWA\_DM\_EXECUTE
- IOWA\_DM\_CREATE
- IOWA\_DM\_DELETE
- IOWA\_DM\_DATA\_PUSH

#### **2.2.4 iowa\_bootstrap\_operation\_t**

```
typedef uint8_t iowa_bootstrap_operation_t;

#define IOWA_BOOTSTRAP_UNDEFINED 0
```

```
#define IOWA_BOOTSTRAP_READ          101
#define IOWA_BOOTSTRAP_WRITE         102
#define IOWA_BOOTSTRAP_DELETE        103
#define IOWA_BOOTSTRAP_DISCOVER      104
#define IOWA_BOOTSTRAP_FINISH        105
#define IOWA_BOOTSTRAP_ADD_SERVER    106
#define IOWA_BOOTSTRAP_REMOVE_SERVER 107
#define IOWA_BOOTSTRAP_ADD_BOOTSTRAP_SERVER 108
#define IOWA_BOOTSTRAP_REMOVE_BOOTSTRAP_SERVER 109
```

This is an enumeration of the following values:

#### **IOWA\_BOOTSTRAP\_UNDEFINED**

No specific LwM2M operation, this should never be used and only serves as a non default operation.

#### **IOWA\_BOOTSTRAP\_READ**

LwM2M Read operation is used to access the value of an Object, Object Instances, and Resources.

#### **IOWA\_BOOTSTRAP\_WRITE**

LwM2M Write operation is used to change the value of an Object, and Object Instances regardless of an existence of the targeted Object Instance(s).

#### **IOWA\_BOOTSTRAP\_DELETE**

LwM2M Delete operation is used to delete any Object Instance or all Instances of any Object including the Security Object (ID:0).

#### **IOWA\_BOOTSTRAP\_DISCOVER**

LwM2M Bootstrap Discover operation is used to discover which LwM2M Objects and Object Instances are supported on a LwM2M Client. In particular, the list of Security Object Instances (ID:0) is reported.

#### **IOWA\_BOOTSTRAP\_FINISH**

LwM2M Finish operation is used to terminate the Bootstrap Sequence.

#### **IOWA\_BOOTSTRAP\_ADD\_SERVER**

Custom LwM2M Bootstrap Add Server operation is used to write the proper Object Instances of Security Object (ID:0) and Server Object (ID:1) to add a LwM2M Server Account.

#### **IOWA\_BOOTSTRAP\_REMOVE\_SERVER**

Custom LwM2M Bootstrap Remove Server operation is used to delete the proper Object Instances of Security Object (ID:0) and Server Object (ID:1) associated to a LwM2M Server Account.

#### **IOWA\_BOOTSTRAP\_ADD\_BOOTSTRAP\_SERVER**

Custom LwM2M Bootstrap Add Bootstrap Server operation is used to write the proper Object Instance of Security Object (ID:0) add a LwM2M Bootstrap Server Account.

#### **IOWA\_BOOTSTRAP\_REMOVE\_BOOTSTRAP\_SERVER**

Custom LwM2M Bootstrap Remove Bootstrap Server operation is used to delete the proper Object Instance of Security Object (ID:0) associated to a LwM2M Bootstrap Server Account.

### **2.2.5 iowa\_lwm2m\_data\_type\_t**

```
typedef uint8_t iowa_lwm2m_data_type_t;

#define IOWA_LWM2M_TYPE_UNDEFINED    0
#define IOWA_LWM2M_TYPE_STRING       1
#define IOWA_LWM2M_TYPE_OPAQUE       2
#define IOWA_LWM2M_TYPE_INTEGER      3
#define IOWA_LWM2M_TYPE_FLOAT        4
#define IOWA_LWM2M_TYPE_BOOLEAN      5
#define IOWA_LWM2M_TYPE_CORE_LINK    6
#define IOWA_LWM2M_TYPE_OBJECT_LINK  7
#define IOWA_LWM2M_TYPE_TIME         8
```

```
#define IOWA_LWM2M_TYPE_UNSIGNED_INTEGER 9
```

This is an enumeration of the following values:

**IOWA\_LWM2M\_TYPE\_UNDEFINED**

No specific data type: it is only used for Executable Resource.

**IOWA\_LWM2M\_TYPE\_STRING**

A UTF-8 string.

**IOWA\_LWM2M\_TYPE\_OPAQUE**

A sequence of binary octets.

**IOWA\_LWM2M\_TYPE\_INTEGER**

An 64-bit signed integer.

**IOWA\_LWM2M\_TYPE\_FLOAT**

A 32 or 64-bit floating point value.

**IOWA\_LWM2M\_TYPE\_BOOLEAN**

An unsigned integer with the value 0 for false and the value 1 for true.

**IOWA\_LWM2M\_TYPE\_CORE\_LINK**

A UTF-8 string representing the relation between resources and links.

**IOWA\_LWM2M\_TYPE\_OBJECT\_LINK**

Reference to an Instance of a given Object.

**IOWA\_LWM2M\_TYPE\_TIME**

A signed integer representing the number of seconds.

**IOWA\_LWM2M\_TYPE\_UNSIGNED\_INTEGER**

An unsigned integer.

## 2.2.6 iowa\_lwm2m\_data\_t

When the Lwm2m Server and the Lwm2m Client exchange data, at the application level, they are presented in `iowa_lwm2m_data_t` structures.

```
typedef struct
{
    uint16_t objectID;
    uint16_t instanceID;
    uint16_t resourceID;
    uint16_t resInstanceID;
    iowa_lwm2m_data_type_t type;
    union
    {
        bool    asBoolean;
        int64_t asInteger;
        double  asFloat;
        struct
        {
            size_t length;
            uint8_t *buffer;
        } asBuffer;
        iowa_lwm2m_object_link_t asObjLink;
    } value;
    int32_t timestamp;
} iowa_lwm2m_data_t;
```

This structure contains the value of a Lwm2m resource along its complete URI.

**objectID**

ID of the Object containing the resource.

**instanceID**

ID of the Object Instance containing the resource.

**resourceID**

ID of the resource.

**resInstanceID**

ID of the resource instance. For single instance resource, this is always **IOWA\_LWM2M\_ID\_ALL**.

**type**

The datatype of the resource.

**value.asBoolean**

The value of the resource when type is **IOWA\_LWM2M\_TYPE\_BOOLEAN**.

**value.asInteger**

The value of the resource when type is **IOWA\_LWM2M\_TYPE\_INTEGER**, **IOWA\_LWM2M\_TYPE\_TIME** or **IOWA\_LWM2M\_TYPE\_UNSIGNED\_INTEGER**.

**value.asFloat**

The value of the resource when type is **IOWA\_LWM2M\_TYPE\_FLOAT**.

**value.asBuffer**

The value of the resource when type is **IOWA\_LWM2M\_TYPE\_CORE\_LINK**, **IOWA\_LWM2M\_TYPE\_STRING**, **IOWA\_LWM2M\_TYPE\_OPAQUE** or **IOWA\_LWM2M\_TYPE\_UNDEFINED**.

**value.asObjLink**

The value of the resource when type is **IOWA\_LWM2M\_TYPE\_OBJECT\_LINK**.

**timestamp**

The timestamp value in seconds. Time is always absolute, and timestamp is present when the value is greater than zero. This can not be negative.

## 2.2.7 iowa\_lwm2m\_object\_link\_t

```
typedef struct {
    uint16_t objectID;
    uint16_t instanceID;
} iowa_lwm2m_object_link_t;
```

## 2.2.8 iowa\_content\_format\_t

```
typedef uint16_t iowa_content_format_t;

#define IOWA_CONTENT_FORMAT_TEXT          0
#define IOWA_CONTENT_FORMAT_OPAQUE       42
#define IOWA_CONTENT_FORMAT_CBOR         60
#define IOWA_CONTENT_FORMAT_SENML_JSON  110
#define IOWA_CONTENT_FORMAT_SENML_CBOR  112
#define IOWA_CONTENT_FORMAT_TLV_OLD      1542
#define IOWA_CONTENT_FORMAT_JSON_OLD     1543
#define IOWA_CONTENT_FORMAT_TLV          11542
#define IOWA_CONTENT_FORMAT_JSON         11543
#define IOWA_CONTENT_FORMAT_UNSET        0xFFFF
```

This is an enumeration of the following values:

**IOWA\_CONTENT\_FORMAT\_TEXT**

Plain text encoding (e.g. "123", "-123.45"). Usable only for single resource encoding.



### **IOWA\_CONTENT\_FORMAT\_OPAQUE**

A sequence of binary octets. Usable only for single resource encoding which data type is **Opaque**.

### **IOWA\_CONTENT\_FORMAT\_CBOR**

CBOR encoding. Usable only for single resource encoding.

### **IOWA\_CONTENT\_FORMAT\_SENML\_JSON**

LwM2M specific SenML JSON encoding. This may not be supported by all Clients. See [iowa\\_client\\_t](#).

### **IOWA\_CONTENT\_FORMAT\_SENML\_CBOR**

LwM2M specific SenML CBOR encoding. This may not be supported by all Clients. See [iowa\\_client\\_t](#).

### **IOWA\_CONTENT\_FORMAT\_TLV\_OLD**

LwM2M specific binary Type-Length-Value format. Usually the most compact one. This one is not anymore used, it only serves as backward compatibility with old LwM2M stack implementation (previous 1.0).

### **IOWA\_CONTENT\_FORMAT\_JSON\_OLD**

LwM2M specific JSON encoding. This may not be supported by all Clients. See [iowa\\_client\\_t](#). This one is not anymore used, it only serves as backward compatibility with old LwM2M stack implementation (previous 1.0).

### **IOWA\_CONTENT\_FORMAT\_TLV**

LwM2M specific binary Type-Length-Value format. Usually the most compact one.

### **IOWA\_CONTENT\_FORMAT\_JSON**

LwM2M specific JSON encoding. This may not be supported by all Clients. See [iowa\\_client\\_t](#).

### **IOWA\_CONTENT\_FORMAT\_UNSET**

Used to reset the encoding to the default one.

## 2.2.9 iowa\_lwm2m\_uri\_t

```
typedef struct
{
    uint16_t objectId;
    uint16_t instanceId;
    uint16_t resourceId;
    uint16_t resInstanceId;
} iowa_lwm2m_uri_t;
```

This structure represents a LwM2M URI.

In the LwM2M resource model, resources are grouped into Objects. These Objects have instances. Hence the URI of a resource is in the form `/{"Object"}/{"Object Instance"}/{"Resource"}`. Moreover some resources, described as *multiple*, can have several instances, leading to URI in the form `/{"Object"}/{"Object Instance"}/{"Resource"}/{"Resource Instance"}`.

#### **objectId**

ID of a LwM2M Object.

#### **instanceId**

ID of the Object Instance.

#### **resourceId**

ID of the resource.

#### **resInstanceId**

ID of the resource instance.

When a segment of the URI is not set, the value of the corresponding field is set to **IOWA\_LWM2M\_ID\_ALL**.

For instance, the URI `/3/0/9` is represented as:

```
iowa_lwm2m_uri_t::objectId = 3
iowa_lwm2m_uri_t::instanceId = 0
iowa_lwm2m_uri_t::resourceId = 9
iowa_lwm2m_uri_t::resInstanceId = IOWA_LWM2M_ID_ALL
```

the URI /5 is represented as:

```
iowa_lwm2m_uri_t::objectId = 5  
iowa_lwm2m_uri_t::instanceId = IOWA_LWM2M_ID_ALL  
iowa_lwm2m_uri_t::resourceId = IOWA_LWM2M_ID_ALL  
iowa_lwm2m_uri_t::resInstanceId = IOWA_LWM2M_ID_ALL
```

Evaluation

## 2.2.10 iowa\_response\_content\_t

This structure contains the response content from `iowa_response_callback_t()` according to its requested *operation*.

```
typedef struct
{
    union
    {
        struct
        {
            size_t dataCount;
            iowa_lwm2m_data_t *dataP;
        } read;
        struct
        {
            uint32_t notificationNumber;
            size_t dataCount;
            iowa_lwm2m_data_t *dataP;
        } observe;
        struct
        {
            size_t dataCount;
            iowa_lwm2m_data_t *dataP;
        } dataPush;
    } details;
} iowa_response_content_t;
```

### details.read

The information related to **IOWA\_DM\_READ** operation.

#### details.read.dataCount

The number of elements in the *details.read.dataP*. This may be 0.

#### details.read.dataP

An array containing the Resource values returned by the Client. This may be nil.

### details.observe

The information related to **IOWA\_DM\_NOTIFY** operation.

#### details.observe.notificationNumber

The notification counter.

#### details.observe.dataCount

The number of elements in the *details.observe.dataP*. This may be 0.

#### details.observe.dataP

An array containing the Resource values returned by the Client. This may be nil.

### details.dataPush

The information related to **IOWA\_DM\_DATA\_PUSH** operation.

#### details.dataPush.dataCount

The number of elements in the *details.dataPush.dataP*. This may be 0.

#### details.dataPush.dataP

An array containing the Resource values send by the Client. This may be nil.

## 2.3 Callbacks

### 2.3.1 iowa\_response\_callback\_t

The device management APIs (`iowa_server_read()`, `iowa_server_observe()`, `iowa_server_write()`, `iowa_server_write_attributes_string()`, `iowa_server_configure_data_push()`, `iowa_bootstrap_server_read()`, `iowa_client_send_sensor_data()`, `iowa_client_send_data()`) are using an `iowa_response_callback_t` to asynchronously return the result of the operation.

```
typedef void(*iowa_response_callback_t) (uint32_t sourceId,
                                         uint8_t operation,
                                         iowa_status_t status,
                                         iowa_response_content_t *contentP,
                                         void *userDataP,
                                         iowa_context_t contextP);
```

#### **sourceId**

The ID of the client targeted by the command for the server APIs (`iowa_server_read()`, `iowa_server_observe()`, `iowa_server_write()`, `iowa_server_write_attributes_string()`, `iowa_server_configure_data_push()`). The ID of the server targeted by the command for the client APIs (`iowa_client_send_sensor_data()`, `iowa_client_send_data()`).

#### **operation**

The type of command matching this result.

#### **status**

The status of the command.

#### **contentP**

The content of the operation. It is nil for **IOWA\_DM\_WRITE**, **IOWA\_DM\_EXECUTE**, **IOWA\_DM\_CREATE**, **IOWA\_DM\_DELETE** and **IOWA\_DM\_WRITE\_ATTRIBUTES** operations.

It is also nil for **IOWA\_DM\_DATA\_PUSH** if operation comes from the client APIs.

It is also nil for **IOWA\_DM\_READ**, **IOWA\_DM\_NOTIFY**, **IOWA\_DM\_DISCOVER** and **IOWA\_DM\_DATA\_PUSH** if the the *status* is different from **IOWA\_205\_COAP\_CONTENT**

#### **userDataP**

A pointer to application specific data. This is a parameter of the matching device management API.

#### **contextP**

The IOWA context on which the device management API was called.

### 2.3.2 iowa\_load\_callback\_t

This callback is called during a context load with the data stored inside the context backup.

```
typedef void(*iowa_load_callback_t) (uint16_t callbackId,  
                                     uint8_t * buffer,  
                                     size_t bufferLength,  
                                     void *userDataP);
```

**callbackId**

The identifier of the callback as passed to `iowa_backup_register_callback()`.

**buffer**

The data loaded from the backup. This can be nil.

**bufferLength**

The length of *buffer* in bytes.

**userDataP**

A pointer to application specific data as passed to `iowa_backup_register_callback()`.

### 2.3.3 iowa\_save\_callback\_t

This callback is called during a context save. It is called first to retrieve the length of the data to save, then a second time with an allocated buffer to fill with the data to save.

```
typedef size_t(*iowa_save_callback_t) (uint16_t callbackId,  
                                       uint8_t * buffer,  
                                       size_t bufferLength,  
                                       void *userDataP);
```

**callbackId**

The identifier of the callback as passed to `iowa_backup_register_callback()`.

**buffer**

A buffer to store the data. This can be nil.

**bufferLength**

The length of *buffer* in bytes.

**userDataP**

A pointer to application specific data as passed to `iowa_backup_register_callback()`.

## 2.4 API

### 2.4.1 iowa\_init

#### Prototype

```
iowa_context_t iowa_init(void * userData);
```

#### Description

`iowa_init()` initializes an IOWA context.

#### Arguments

##### ***userData***

Pointer to application-specific data. This is passed as argument to the Communication Abstraction Interface functions. This can be nil.

#### Return Value

An `iowa_context_t` in case of success or NULL in case of memory allocation error.

#### Header File

`iowa.h`

## 2.4.2 iowa\_step

### Prototype

```
iowa_status_t iowa_step(iowa_context_t contextP,  
                        int32_t timeout);
```

### Description

`iowa_step()` runs the stack engine during the specified time.

### Arguments

#### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### **timeout**

The allowed time to run in seconds.

### Return Value

#### **IOWA\_COAP\_NO\_ERROR**

success.

#### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- `iowa_system_gettime()` returned an error.

#### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

one of the `iowa_system_connection...` functions returned an error.

LwM2M Client only: when the Client failed to connect to any Server. (`iowa_event_callback_t` is called with a **IOWA\_EVENT\_REG\_FAILED** event.)

### Header File

`iowa.h`

### Notes

If `timeout` is a negative value:

- `iowa_step()` will return only in case of error.
- `iowa_system_connection_select()` will be called with **INT32\_MAX**.

For LwM2M Clients: if `iowa_step()` returns an **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE** error because it is not registered to any LwM2M Server, subsequent call to `iowa_step()`, will retry to register to the known LwM2M Servers.



### 2.4.3 iowa\_flush\_before\_pause

#### Prototype

```
iowa_status_t iowa_flush_before_pause(iowa_context_t contextP,  
                                     int32_t duration,  
                                     uint32_t *delayP);
```

#### Description

`iowa_flush_before_pause()` is used to inform the stack that the device will pause. `iowa_flush_before_pause()` performs all the pending and required operations of the stack engine before returning.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **duration**

The duration of the planned pause in seconds.

##### **delayP**

The delay before the next IOWA scheduled operation.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- *duration* is negative.
- for a LwM2M Client: *duration* is longer than one of the LwM2M Server registration lifetime.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- `iowa_system_gettime()` returned an error.

##### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

one of the `iowa_system_connection...` functions returned an error.

#### Header File

`iowa.h`

#### Notes

In LwM2M Client mode: if no server are configured, the function returns immediately with no error.

A LwM2M Server should never stop. For a LwM2M Server, `iowa_flush_before_pause()` will just wait for all pending CoAP transactions to finish.

## 2.4.4 iowa\_stop

### Prototype

```
void iowa_stop(iowa_context_t contextP);
```

### Description

`iowa_stop()` stops the stack engine and make `iowa_step()` return immediately.

### Arguments

#### ***contextP***

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

### Return Value

None.

### Header File

`iowa.h`

## 2.4.5 iowa\_close

### Prototype

```
void iowa_close(iowa_context_t contextP);
```

### Description

`iowa_close()` closes an IOWA context.

### Arguments

#### ***contextP***

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

### Return Value

None.

### Header File

`iowa.h`

## 2.4.6 iowa\_save\_context

### Prototype

```
iowa_status_t iowa_save_context(iowa_context_t contextP);
```

### Description

`iowa_save_context()` saves the current IOWA context.

### Arguments

#### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

### Return Value

#### **IOWA\_COAP\_NO\_ERROR**

success.

#### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- the `iowa_system_store_context()` function returned an error.

### Header File

`iowa.h`

### Notes

Currently, this API is only for a LwM2M Client.

If IOWA is built with the **IOWA\_STORAGE\_CONTEXT\_AUTOMATIC\_BACKUP** flag, the context will be automatically saved:

- After a LwM2M Bootstrap Server or a LwM2M Server were added.
- After a successful Bootstrap procedure.
- After a LwM2M Server operation on resources related to Server Accounts.

## 2.4.7 iowa\_save\_context\_snapshot

### Prototype

```
iowa_status_t iowa_save_context_snapshot(iowa_context_t contextP);
```

### Description

`iowa_save_context_snapshot()` saves the current IOWA context with runtime information, observations and attributes.

### Arguments

#### ***contextP***

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

### Return Value

#### **IOWA\_COAP\_NO\_ERROR**

success.

#### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- the `iowa_system_store_context()` function returned an error.

### Header File

`iowa.h`

### Notes

Currently, this API is only for a LwM2M Client.

## 2.4.8 iowa\_load\_context

### Prototype

```
iowa_status_t iowa_load_context(iowa_context_t contextP);
```

### Description

`iowa_load_context()` loads a saved IOWA context.

### Arguments

#### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

### Return Value

#### **IOWA\_COAP\_NO\_ERROR**

success.

#### **IOWA\_COAP\_404\_NOT\_FOUND**

either:

- no previous saved IOWA context found.
- In LwM2M client mode: no Security or Server objects found, the function `iowa_client_configure` must be called first.

#### **IOWA\_COAP\_409\_CONFLICT**

`contextP` isn't in init state.

#### **IOWA\_COAP\_422\_UNPROCESSABLE\_ENTITY**

either:

- failed to decode the buffer retrieved from `iowa_system_retrieve_context()` function.
- context version isn't present or is incorrect.

#### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- the `iowa_system_retrieve_context()` function returned an error.

### Header File

`iowa.h`

### Notes

Currently, this API is only for a LwM2M Client.

## 2.4.9 iowa\_backup\_register\_callback

### Prototype

```
iowa_status_t iowa_backup_register_callback(iowa_context_t contextP,
                                           uint16_t id,
                                           iowa_save_callback_t saveCallback,
                                           iowa_load_callback_t loadCallback,
                                           void *userDataP);
```

### Description

`iowa_backup_register_callback()` registers context callbacks from IOWA.

### Arguments

#### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### **id**

The id of the callback. Must be greater than 0xF000.

#### **saveCallback**

The function called during context saving.

#### **loadCallback**

The function called during context loading.

#### **userDataP**

Pointer to application-specific data. This is passed as argument to the callback. This can be nil.

### Return Value

#### **IOWA\_COAP\_NO\_ERROR**

success.

#### **IOWA\_COAP\_403\_FORBIDDEN**

either:

- id is incorrect.
- one of the callback is NULL.

#### **IOWA\_COAP\_409\_CONFLICT**

id is already used.

#### **IOWA\_COAP\_422\_UNPROCESSABLE\_ENTITY**

failed to decode the buffer retrieved from `iowa_system_retrieve_context()` function.

#### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

### Header File

`iowa.h`

### Notes

Currently, this API is only for a LwM2M Client.

## 2.4.10 iowa\_backup\_deregister\_callback

### Prototype

```
void iowa_backup_deregister_callback(iowa_context_t contextP,  
                                     uint16_t id);
```

### Description

`iowa_backup_deregister_callback()` deregisters context callbacks from IOWA.

### Arguments

#### ***contextP***

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### ***id***

The id of the callback. Must be greater than 0xF000.

### Return Value

None.

### Header File

`iowa.h`

### Notes

Currently, this API is only for a LwM2M Client.



## 2.4.11 iowa\_connection\_closed

### Prototype

```
void iowa_connection_closed(iowa_context_t contextP,  
                           void *connP);
```

### Description

`iowa_connection_closed()` informs IOWA that a connection was closed by an external event (e.g. peer disconnection).

### Arguments

#### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### **connP**

The closed connection of the same user-defined type as the one returned by `iowa_system_connection_open()`.

### Return Value

None.

### Header File

iowa.h

## 3 | Client Mode API Reference

The functions explained below are defined inside the file `include/iowa_client.h` and the Objects folder `include/objects`.

### 3.1 Client pseudo code

```
#include "iowa_client.h"
#include "iowa_ipso.h"

int main(int argc,
         char *argv[])
{
    iowa_context_t iowaH;
    iowa_status_t result;
    iowa_device_info_t devInfo;
    iowa_sensor_t sensorId;

    /*****
    * Initialization
    */

    iowaH = iowa_init(NULL);

    devInfo.manufacturer = "IOTEROP";
    devInfo.deviceType = "Example device";
    devInfo.modelNumber = "1";
    devInfo.serialNumber = NULL;
    devInfo.hardwareVersion = NULL;
    devInfo.softwareVersion = NULL;
    devInfo.optFlags = 0;
    result = iowa_client_configure(iowaH, "IOWA_Sample_Client", devInfo, NULL);

    result = iowa_client_IPSO_add_sensor(iowaH,
                                         IOWA_IPSO_VOLTAGE, 12.0,
                                         "V", "Test DC", 0.0, 0.0,
                                         &sensorId);

    result = iowa_client_add_server(iowaH, 1234, "coap://localhost:5683", 0, 0,
                                    IOWA_SEC_NONE);

    /*****
    * "Main loop"
    */

    while (result == IOWA_COAP_NO_ERROR)
    {
        float sensorValue;

        result = iowa_step(iowaH, 5);

        sensorValue = read_battery_voltage();
        result = iowa_client_IPSO_update_value(iowaH,
                                                sensorId,
```

```
        sensorValue);  
  
    }  
  
    iowa_client_IPSO_remove_sensor(iowaH, sensorId);  
    iowa_close(iowaH);  
  
    return 0;  
}
```

Evaluation

## 3.2 Data types

### 3.2.1 iowa\_device\_info\_t

```
typedef struct
{
    const char *manufacturer;
    const char *deviceType;
    const char *modelName;
    const char *serialNumber;
    const char *hardwareVersion;
    const char *firmwareVersion;
    const char *softwareVersion;
    const char *msisdn;
    uint16_t    optFlags;
    const char *utcOffsetP;
    const char *timezoneP;
    iowa_client_time_update_callback_t    dataTimeUpdateCallback;
    iowa_client_factory_reset_callback_t  factoryResetCallback;
    void                                *callbackUserDataP;
} iowa_device_info_t;
```

The `iowa_device_info_t` structure exists only for the sake of the readability of `iowa_client_configure()`. It contains pointers to nil-terminated strings described below. As all these information are optional in a LwM2M Client, these pointers can be nil. The LwM2M standard does not mandate any format for these strings. They are manufacturer specific.

#### **manufacturer**

A human readable manufacturer name.

#### **deviceType**

The type of the device.

#### **modelName**

The number of the model.

#### **serialNumber**

The serial number of the device.

#### **hardwareVersion**

The current version of the device hardware.

#### **firmwareVersion**

The current version of the device firmware.

#### **softwareVersion**

The current version of the device software.

#### **msisdn**

The phone number of the device.

#### **optFlags**

Flags used to enable optional features. This value is a combination of:

- **IOWA\_DEVICE\_RSC\_BATTERY**: enables the battery level and status exposed in the [Device Object][Device Object]. To update battery level you need to call `iowa_client_device_update_battery()`.
- **IOWA\_DEVICE\_RSC\_POWER\_SOURCE**: enables the power sources information in the [Device Object][Device Object]. To add any new power source you need to call `iowa_client_add_device_power_source()`.
- **IOWA\_DEVICE\_RSC\_CURRENT\_TIME**: enables the use of current time in the [Device Object][Device Object] (default value: 0).
- **IOWA\_DEVICE\_RSC\_UTC\_OFFSET**: enables the use of UTC offset in the [Device Object][Device Object] (default value: `utcOffsetP`).
- **IOWA\_DEVICE\_RSC\_TIMEZONE**: enables the use of timezone in the [Device Object][Device Object] (de-

fault value: 0). To update time information (current time, UTC offset, timezone) you need to call `iowa_client_update_device_time_information()`

#### **utcOffsetP**

Indicates the UTC offset currently in effect for this LwM2M Device. It should be in the ISO 8601 format (UTC+X).

#### **timezoneP**

Indicates in which time zone the LwM2M Device is located, in IANA Timezone (TZ) database format.

#### **dataTimeUpdateCallback**

The callback called when the time information is updated by the LwM2M Server.

#### **factoryResetCallback**

The callback called on a Factory Reset.

#### **callbackUserDataP**

Passed as argument to the callbacks `dataTimeUpdateCallback` and `factoryResetCallback`.

### 3.2.2 iowa\_event\_type\_t

```
typedef enum
{
    IOWA_EVENT_UNDEFINED = 0,
    IOWA_EVENT_REG_UNREGISTERED,
    IOWA_EVENT_REG_REGISTERING,
    IOWA_EVENT_REG_REGISTERED,
    IOWA_EVENT_REG_UPDATING,
    IOWA_EVENT_REG_FAILED,
    IOWA_EVENT_REG_UPDATE_FAILED,
    IOWA_EVENT_BS_PENDING,
    IOWA_EVENT_BS_FINISHED,
    IOWA_EVENT_BS_FAILED,
    IOWA_EVENT_OBSERVATION_STARTED,
    IOWA_EVENT_OBSERVATION_NOTIFICATION,
    IOWA_EVENT_OBSERVATION_CANCELED,
    IOWA_EVENT_OBJECT_INSTANCE_CREATED,
    IOWA_EVENT_OBJECT_INSTANCE_DELETED,
    IOWA_EVENT_EVALUATION_PERIOD,
    IOWA_EVENT_READ
} iowa_event_type_t;
```

The `iowa_event_type_t` contains the possible events that can be reported by the IOWA stack.

### 3.2.3 iowa\_event\_t

```
typedef struct
{
    iowa_event_type_t eventType;
    uint16_t serverShortId;
    union
    {
        struct
        {
            uint32_t lifetime;
        } registration;
        struct
        {
            iowa_sensor_t sensorId;
        }
    }
}
```

```

        uint16_t resourceId;
        uint32_t minPeriod;
        uint32_t maxPeriod;
        uint32_t minEvalPeriod;
        uint32_t maxEvalPeriod;
    } observation;
    struct
    {
        iowa_lwm2m_uri_t * uriP;
    } objectInstance;
    struct
    {
        iowa_lwm2m_uri_t * uriP;
        uint32_t minEvalPeriod;
        uint32_t maxEvalPeriod;
    } evalPeriod;
    struct
    {
        iowa_sensor_t sensorId;
    } sensor;
    } details;
} iowa_event_t;

```

### eventType

the event type.

### serverShortId

the short server ID of the LwM2M Server generating this event.

### details

the details of the event.

#### details::registration

filled when the event is of type **IOWA\_EVENT\_REG\_UNREGISTERED**, **IOWA\_EVENT\_REG\_REGISTERING**, **IOWA\_EVENT\_REG\_REGISTERED**, **IOWA\_EVENT\_REG\_UPDATING**, **IOWA\_EVENT\_REG\_FAILED** or **IOWA\_EVENT\_REG\_UPDATE\_FAILED**.

#### details::registration::lifetime

the lifetime of the registration to the LwM2M Server generating this event.

#### details::observation

filled when the event is of type **IOWA\_EVENT\_OBSERVATION\_STARTED**, **IOWA\_EVENT\_OBSERVATION\_NOTIFICATION**, or **IOWA\_EVENT\_OBSERVATION\_CANCELED**

#### details::observation::sensorId

the ID of the sensor under observation.

#### details::observation::resourceId

the ID of the specific resource under observation of the sensor. This may be **IOWA\_LWM2M\_ID\_ALL**.

#### details::observation::minPeriod

the minimum time in seconds to wait between notifications for the observation. If not set the minPeriod is to 0.

#### details::observation::maxPeriod

the maximum time in seconds to wait between notifications for the observation. If not set the maxPeriod is to **UINT32\_MAX**.

#### details::observation::minEvalPeriod

the minimum sample time in seconds for the observed sensor in LwM2M 1.1 or later. If not set the minEvalPeriod is to 0.

#### details::observation::maxEvalPeriod

the maximum sample time in seconds for the observed sensor in LwM2M 1.1 or later. If not set the maxEvalPeriod is to **UINT32\_MAX**.

#### details::instance

filled when the event is of type **IOWA\_EVENT\_OBJECT\_INSTANCE\_CREATED** or

## IOWA\_EVENT\_OBJECT\_INSTANCE\_DELETED.

### details::instance::uri

a pointer to the `iowa_lwm2m_uri_t` of the instance that has been created or deleted.

### details::evalPeriod

filled when the event is of type **IOWA\_EVENT\_EVALUATION\_PERIOD**. Available when the flag `IOWA_LWM2M_VERSION_1_1` is set.

### details::evalPeriod::uriP

a pointer to the `iowa_lwm2m_uri_t` of the uri where evaluation period has been set.

### details::evalPeriod::minEvalPeriod

the minimum sample time in seconds for the concerned uri. If the Lwm2M Server unsets it or does not set it, the value is 0.

### details::evalPeriod::maxEvalPeriod

the maximum sample time in seconds for the concerned uri. If the Lwm2M Server unsets it or does not set it, the value is `UINT32_MAX`.

### details::sensor

filled when the event is of type **IOWA\_EVENT\_READ**.

### details::sensor::sensorId

the ID of the sensor targeted by a Read operation from the Lwm2M Server.

The `iowa_event_t` is used by `iowa_event_callback_t` when an event occurred on a client. These events are described by `iowa_event_type_t`.

The IOWA stack handles the minimum and maximum observation periods. They are provided in `iowa_event_t` as an information for the application. Embedded devices may use this information to tune their measurement or sleeping schedule.

The IOWA stack does not handle the minimum and maximum evaluation observation periods. They are provided in `iowa_event_t` as sample times for the application. Embedded devices may use those sample times to tune their measurement or sleeping schedule.

## 3.2.4 iowa\_device\_time\_info\_t

```
typedef struct
{
    uint16_t    flags;
    uint32_t    currentTime;
    const char *utcOffsetP;
    const char *timezoneP;
} iowa_device_time_info_t;
```

### flags

Flags used to enable optional time information. This value is a combination of:

- **IOWA\_DEVICE\_RSC\_CURRENT\_TIME** : current time has a new current time value from server
- **IOWA\_DEVICE\_RSC\_UTC\_OFFSET** : current time has a new UTC offset value from server
- **IOWA\_DEVICE\_RSC\_TIMEZONE** : current time has a new timezone value from server

### currentTime

Current UNIX time of the Lwm2M Client in seconds.

### utcOffsetP

Indicates the UTC offset currently in effect for this Lwm2M Device. It should be in the ISO 8601 format (UTC+X). Could be nil, if not UTC offset is enable.

### timezoneP

Indicates in which time zone the LwM2M Device is located, in IANA Timezone (TZ) database format. Could be nil, if not Timezone is enable.

### 3.2.5 iowa\_ipso\_timed\_value\_t

```
typedef struct
{
    float value;
    int32_t timestamp;
} iowa_ipso_timed_value_t;
```

#### value

The timestamped value.

#### timestamp

The timestamp associated to the value in seconds. This can not be negative.

### 3.2.6 iowa\_sensor\_t

This must be treated as an opaque type. It is internally mapped to a 32-bit unsigned integer.

#### 3.2.6.1 Special Values

##### IOWA\_INVALID\_SENSOR\_ID

Used to indicate an error by APIs returning an `iowa_sensor_t`.

##### IOWA\_DEVICE\_TIME\_SENSOR\_ID

The sensor ID of the Current Time of the device. This is internally mapped to the resource 13 in the [Device Object][Device Object].

### 3.2.7 iowa\_lwm2m\_resource\_desc\_t

This structure contains the description of a LwM2M resource.

```
typedef struct {
    uint16_t id;
    iowa_lwm2m_data_type_t type;
    uint8_t operations;
    uint8_t flags;
} iowa_lwm2m_resource_desc_t;
```

#### id

ID of the resource.

#### type

The datatype of the resource.

#### operations

The operations allowed on the resource.

This is a mask of values **IOWA\_OPERATION\_READ**, **IOWA\_OPERATION\_WRITE** and **IOWA\_OPERATION\_EXECUTE**.

#### flags

The flags of the resource.

This is a mask of values **IOWA\_RESOURCE\_FLAG\_NONE**, **IOWA\_RESOURCE\_FLAG\_OPTIONAL**, **IOWA\_RESOURCE\_FLAG\_MANDATORY**, **IOWA\_RESOURCE\_FLAG\_MULTIPLE**, and **IOWA\_RESOURCE\_FLAG\_ASYNCHRONOUS**.



### 3.2.8 iowa\_sensor\_uri\_t

This structure describes a sensor URI.

```
typedef struct
{
    iowa_sensor_t id;
    uint16_t resourceId;
} iowa_sensor_uri_t;
```

***id***

ID of the object.

***resourceId***

The ID of the resource. This can be **IOWA\_LWM2M\_ID\_ALL**.

## 3.3 Callbacks

### 3.3.1 iowa\_event\_callback\_t

This is the event callback, called when an event such as registration update or unregister occurred.

```
typedef void(*iowa_event_callback_t) (iowa_event_t* eventP,  
                                     void * userData,  
                                     iowa_context_t contextP);
```

#### **eventP**

The event stored in a structure.

#### **userData**

A pointer to application specific data. This is a parameter of `iowa_init()`.

#### **contextP**

The IOWA context on which `iowa_client_configure()` was called.

### 3.3.2 iowa\_client\_time\_update\_callback\_t

This callback is called when time information are updated by server.

```
typedef void(*iowa_client_time_update_callback_t) (iowa_device_time_info_t *timeInfoP,  
                                                  void *userDataP,  
                                                  iowa_context_t contextP);
```

#### **timeInfoP**

Current device time information.

#### **userDataP**

A pointer to application specific data. This is the parameter of `iowa_client_configure()`.

#### **contextP**

The IOWA context on which `iowa_client_configure()` was called.

### 3.3.3 iowa\_client\_factory\_reset\_callback\_t

This callback is called when a factory reset is requested.

```
typedef void(*iowa_client_factory_reset_callback_t) (void *userDataP,  
                                                    iowa_context_t contextP);
```

#### **userDataP**

A pointer to application specific data. This is the parameter of `iowa_client_configure()`.

#### **contextP**

The IOWA context on which `iowa_client_configure()` was called.

### 3.3.4 iowa\_RWE\_callback\_t

This callback is called when a Read, Write or Execute operation is performed on a resource of a custom LwM2M Object.

```
typedef iowa_status_t(*iowa_RWE_callback_t) (iowa_dm_operation_t operation,  
                                             iowa_lwm2m_data_t * dataP,  
                                             size_t numData,  
                                             void * userData,  
                                             iowa_context_t contextP);
```

### **operation**

The operation to perform on the resource among **IOWA\_DM\_READ**, **IOWA\_DM\_WRITE** and **IOWA\_DM\_EXECUTE**.

### **dataP**

An array of the URIs of the targeted resources.

For a Write operation, it also contains the value to write.

For a Read operation, the result is to be stored in this.

### **numData**

Number of resources in dataP.

### **userData**

A pointer to application specific data. This is the parameter of `iowa_client_add_custom_object()`.

### **contextP**

The IOWA context on which `iowa_client_add_custom_object()` was called.

#### **3.3.4.1 Notes**

- Before calling this callback, the IOWA stack performs checks on the resource existence and its allowed operations. In case of a Write operation, the data type is also checked for conformance.
- The Lwm2M Execute operation may have parameters. If so, they are provided as a string in *dataP*.
- After **IOWA\_DM\_READ** operation, the callback is called with **IOWA\_DM\_FREE** operation to permit the deallocation of memory that may have been allocated by the callback previously.

#### **3.3.5 iowa\_CD\_callback\_t**

This callback is called when a Create or Delete operation is performed on an instance of a custom Lwm2M Object.

```
typedef iowa_status_t(*iowa_CD_callback_t) (iowa_dm_operation_t operation,
                                           uint16_t objectID,
                                           uint16_t instanceID,
                                           void * userData,
                                           iowa_context_t contextP);
```

### **operation**

The operation to perform on the resource among **IOWA\_DM\_CREATE** and **IOWA\_DM\_DELETE**.

### **objectID**

The ID of the targeted Object.

### **instanceID**

The ID of the targeted instance.

### **userData**

A pointer to application specific data. This is the parameter of `iowa_client_add_custom_object()`.

### **contextP**

The IOWA context on which `iowa_client_add_custom_object()` was called.

#### **3.3.6 iowa\_RI\_callback\_t**

This callback is called to retrieve the list of current resource instance IDs for a multiple resource.

```
typedef iowa_status_t(*iowa_RI_callback_t) (uint16_t objectID,
                                           uint16_t instanceID,
                                           uint16_t resourceID,
                                           uint16_t * nbResInstanceP,
                                           uint16_t ** resInstanceArrayP,
                                           void * userData,
                                           iowa_context_t contextP);
```

***objectID***

The ID of the Object the resource belongs to.

***instanceID***

The ID of the Object Instance the resource belongs to.

***resourceID***

The ID of the targeted resource.

***nbResInstanceP***

Used to store the number of elements in resInstanceArrayP.

***resInstanceArrayP***

Used to store an array containing the resource instances IDs. This array will be freed by the caller by calling `iowa_system_free()`.

***userData***

A pointer to application specific data. This is the parameter of `iowa_client_add_custom_object()`.

***contextP***

The IOWA context on which `iowa_client_add_custom_object()` was called.

Evaluation

## 3.4 API

### 3.4.1 iowa\_client\_configure

#### Prototype

```
iowa_status_t iowa_client_configure(iowa_context_t contextP,  
                                   const char * identity,  
                                   iowa_device_info_t * infoP,  
                                   iowa_event_callback_t eventCb);
```

#### Description

`iowa_client_configure()` sets the information of the LwM2M Client.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **identity**

The unique identity of the LwM2M Client as a nil-terminated string.

##### **infoP**

The optional information of the LwM2M Client. This can be nil.

##### **eventCb**

The callback called when an event occurred. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **COAP\_400\_BAD\_REQUEST**

either:

- *identity* is nil or empty and **LWM2M\_VERSION\_1\_1\_SUPPORT** is not set.
- the maximum length of *infoP->msisdn* is 15 digits.
- *infoP->msisdn* is not nil, but **IOWA\_SMS\_SUPPORT** is not defined.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

the client was already configured in this context. To reconfigure the client, close then reopen a fresh IOWA context with `iowa_close()` and `iowa_init()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`iowa_client.h`

#### Notes

The nil-terminated strings pointed by the fields of *infoP* are not duplicated nor freed by IOWA. Make sure they are available until `iowa_close()` is called. It is advised to use static strings.

The LwM2M Client information are exposed to the LwM2M Server through the Resources of the [Device Object][Device Object] (ID: 3). The following table explained the mapping:

Resource ID	Resource Name	API
0	Manufacturer	<i>manufacturer</i> field of the <i>iowa_device_info_t</i> structure.
1	Model Number	<i>modelName</i> field of the <i>iowa_device_info_t</i> structure.
2	Serial Number	<i>serialNumber</i> field of the <i>iowa_device_info_t</i> structure.
3	Firmware Version	<i>firmwareVersion</i> field of the <i>iowa_device_info_t</i> structure.
4	Reboot	
5	Factory Reset	
6	Available Power Sources	Set the flag <b>IOWA_DEVICE_RSC_POWER_SOURCE</b> in <i>optFlags</i> field of the structure <i>iowa_device_info_t</i> . Then use <i>iowa_client_..._device_power_source</i> to control it.
7	Power Source Voltage	Set the flag <b>IOWA_DEVICE_RSC_POWER_SOURCE</b> in <i>optFlags</i> field of the structure <i>iowa_device_info_t</i> . Then use <i>iowa_client_..._device_power_source</i> to control it.
8	Power Source Current	Set the flag <b>IOWA_DEVICE_RSC_POWER_SOURCE</b> in <i>optFlags</i> field of the structure <i>iowa_device_info_t</i> . Then use <i>iowa_client_..._device_power_source</i> to control it.
9	Battery Level	Set the flag <b>IOWA_DEVICE_RSC_BATTERY</b> in <i>optFlags</i> field of the structure <i>iowa_device_info_t</i> . Then use <i>iowa_client_device_update_battery</i> to control it.
10	Memory Free	Not exposed by IOWA.
11	Error Code	Use <i>iowa_client_..._device_error_code</i> to control it.
12	Reset Error Code	Set the flag <b>IOWA_DEVICE_RSC_RESET_ERROR</b> in <i>optFlags</i> field of the structure <i>iowa_device_info_t</i> .
13	Current Time	Set the flag <b>IOWA_DEVICE_RSC_CURRENT_TIME</b> in <i>optFlags</i> field of the structure <i>iowa_device_info_t</i> . Then use <i>iowa_client_update_device_time_information</i> to control it.
14	UTC Offset	<i>utcOffsetP</i> field and set the flag <b>IOWA_DEVICE_RSC_UTC_OFFSET</b> in <i>optFlags</i> field of the structure <i>iowa_device_info_t</i> . Then use <i>iowa_client_update_device_time_information</i> to control it.
15	Timezone	<i>timezoneP</i> field and set the flag <b>IOWA_DEVICE_RSC_TIMEZONE</b> in <i>optFlags</i> field of the structure <i>iowa_device_info_t</i> . Then use <i>iowa_client_update_device_time_information</i> to control it.
16	Supported Binding and Modes	Cannot be updated directly but depends on the Server URI schema.
17	Device Type	<i>deviceType</i> field of the <i>iowa_device_info_t</i> structure.
18	Hardware Version	<i>hardwareVersion</i> field of the <i>iowa_device_info_t</i> structure.
19	Software Version	<i>softwareVersion</i> field of the <i>iowa_device_info_t</i> structure.
20	Battery Status	Set the flag <b>IOWA_DEVICE_RSC_BATTERY</b> in <i>optFlags</i> field of the structure <i>iowa_device_info_t</i> . Then use <i>iowa_client_device_update_battery</i> to control it.
21	Memory Total	Not exposed by IOWA.

Resource ID	Resource Name	API
22	ExtDevInfo	Not exposed by IOWA.

Evaluation

### 3.4.2 iowa\_client\_new\_incoming\_connection

#### Prototype

```
iowa_status_t iowa_client_new_incoming_connection(iowa_context_t contextP,
                                                  iowa_connection_type_t type,
                                                  void *connP,
                                                  bool isSecure);
```

#### Description

`iowa_client_new_incoming_connection()` informs the stack of a new incoming connection.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **type**

The type of the new connection. See `iowa_connection_type_t`.

##### **connP**

The new connection of the same user-defined type as the one returned by `iowa_system_connection_open()`.

##### **isSecure**

Set to `true` if the security must be enabled on this connection.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- the connection type is not supported. Make sure to verify the corresponding `IOWA_..._SUPPORT` flag has been enabled during IOWA build.
- `isSecure` is true, but no security layer has been built. Make sure to verify the corresponding `IOWA_SECURITY_LAYER_..._` flag has been enabled during IOWA build.

#### Header File

`iowa_client.h`

#### Notes

`iowa_client_new_incoming_connection()` can only be called when IOWA is built with the flag `LWM2M_CLIENT_INCOMING_CONNECTION_SUPPORT`.



### 3.4.3 iowa\_client\_add\_bootstrap\_server

#### Prototype

```
iowa_status_t iowa_client_add_bootstrap_server(iowa_context_t contextP,  
                                              const char *uri,  
                                              iowa_security_mode_t securityMode);
```

#### Description

`iowa_client_add_bootstrap_server()` declares a new LwM2M Bootstrap Server for the LwM2M Client to connect to.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime. The context MUST be configured with `iowa_client_configure()` to add a bootstrap server.

##### **uri**

The URI to reach this bootstrap server as a nil-terminated string e.g. "coaps://[::1]:5684", "coap://lwm2m.example.org:5683" or "sms://+331020304050".

##### **securityMode**

The security mode to use when connecting to this LwM2M Bootstrap Server. See `[iowa_security_mode_t][iowa_security_mode_t]`.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

`uri` is nil.

##### **IOWA\_COAP\_403\_FORBIDDEN**

a bootstrap server is already configured. To reconfigure the LwM2M Bootstrap Server, call first `iowa_client_remove_bootstrap_server`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

client is not configured. Call first `iowa_client_configure()`.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

`uri` is invalid. For example, if the transport is not supported or if `uri` does not match `securityMode`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`iowa_client.h`

#### Notes

`iowa_client_add_bootstrap_server()` can only be called when IOWA is built with the flag **LWM2M\_BOOTSTRAP**.

`uri` is duplicated internally by IOWA and can be freed by the caller.

Only one bootstrap server can be configured.

### 3.4.4 iowa\_client\_remove\_bootstrap\_server

#### Prototype

```
iowa_status_t iowa_client_remove_bootstrap_server(iowa_context_t contextP);
```

#### Description

`iowa_client_remove_bootstrap_server()` removes a LwM2M Bootstrap Server added by `iowa_client_add_bootstrap_server()` from the LwM2M Client.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

no bootstrap server is configured. `iowa_client_add_bootstrap_server()` was not called before, or failed.

#### Header File

`iowa_client.h`

#### Notes

`iowa_client_remove_bootstrap_server()` can only be called when IOWA is built with the flag **LWM2M\_BOOTSTRAP**.

### 3.4.5 iowa\_client\_set\_bootstrap\_server\_hold\_off

#### Prototype

```
iowa_status_t iowa_client_set_bootstrap_server_hold_off(iowa_context_t contextP,  
int32_t holdOff);
```

#### Description

`iowa_client_set_bootstrap_server_hold_off()` sets the Bootstrap Hold Off time.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **holdOff**

The Hold Off time.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

`holdOff` is negative.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

no bootstrap server is configured. `iowa_client_add_bootstrap_server()` was not called before, or failed.

#### Header File

`iowa_client.h`

#### Notes

`iowa_client_set_bootstrap_server_hold_off()` can only be called when IOWA is built with the flag **LWM2M\_BOOTSTRAP**.

### 3.4.6 iowa\_client\_get\_bootstrap\_server\_coap\_peer

#### Prototype

```
iowa_coap_peer_t * iowa_client_get_bootstrap_server_coap_peer(iowa_context_t contextP);
```

#### Description

`iowa_client_get_bootstrap_server_coap_peer()` returns the CoAP peer associated to a LwM2M Bootstrap Server.

#### Arguments

##### ***contextP***

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### Return Value

A pointer to the `iowa_coap_peer_t` associated to the LwM2M Bootstrap Server.

This pointer may be nil if IOWA did not yet initiate, or has finished, the Bootstrap process.

#### Header File

`iowa_client.h`

#### Notes

`iowa_client_get_bootstrap_server_coap_peer()` can only be called when IOWA is built with the flag **LWM2M\_BOOTSTRAP**.

### 3.4.7 iowa\_client\_add\_server

#### Prototype

```
iowa_status_t iowa_client_add_server(iowa_context_t contextP,
                                     uint16_t shortID,
                                     const char *uri,
                                     uint32_t lifetime,
                                     uint16_t configFlags,
                                     iowa_security_mode_t securityMode);
```

#### Description

`iowa_client_add_server()` declares a new LwM2M Server for the LwM2M Client to connect to.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime. The context MUST be configured with `iowa_client_configure()` to add a server.

##### **shortID**

The ID assigned to this server. This cannot be zero nor **IOWA\_LWM2M\_ID\_ALL** nor an existing one.

##### **uri**

The URI to reach this server as a nil-terminated string e.g. “coaps://[::1]:5684”, “coap://lwm2m.example.org:5683” or “sms://+331020304050”.

##### **lifetime**

The lifetime in seconds of the registration to this server.

##### **configFlags**

A bit-mask of configuration flags for this LwM2M Server.

##### **securityMode**

The security mode to use when connecting to this LwM2M Server. See `[iowa_security_mode_t][iowa_security_mode_t]`.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

*uri* is nil.

##### **IOWA\_COAP\_403\_FORBIDDEN**

*shortID* is either zero, **IOWA\_LWM2M\_ID\_ALL** or already in use.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

client is not configured. Call first `iowa_client_configure()`.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

*uri* is invalid. For example, if the transport is not supported or if *uri* does not match *securityMode*.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`iowa_client.h`

#### Notes

If *lifetime* is set to zero, the registration lifetime is set to a default value of:

- 30 days (2,592,000 seconds) for LoRaWAN transport
- 24 hours (86,400 seconds) for other transports (UDP, TCP, SMS ...)

*uri* is duplicated internally by IOWA and can be freed by the caller.

*configFlags* is a combination of the following:

- **IOWA\_LWM2M\_QUEUE\_MODE**: Enable LwM2M Queue Mode for this LwM2M Server.

Evaluation

### 3.4.8 iowa\_client\_remove\_server

#### Prototype

```
iowa_status_t iowa_client_remove_server(iowa_context_t contextP,  
                                         uint16_t shortID);
```

#### Description

`iowa_client_remove_server()` removes a LwM2M Server added by `iowa_client_add_server()` from the LwM2M Client.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **shortID**

The ID assigned to this server or `IOWA_LWM2M_ID_ALL`.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_403\_FORBIDDEN**

*shortID* is zero.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

*shortID* does not match any known server.

#### Header File

`iowa_client.h`

### 3.4.9 iowa\_client\_set\_server\_msisdn

#### Prototype

```
iowa_status_t iowa_client_set_server_msisdn(iowa_context_t contextP,  
                                             uint16_t shortID,  
                                             const char * msisdn);
```

#### Description

`iowa_client_set_server_msisdn()` sets the MSISDN of a previously added LwM2M Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **shortID**

the Short ID assigned to a LwM2M Server.

##### **msisdn**

the MSISDN to reach this Server e.g. "0102030405" or "+33102030405". This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

the maximum length of `msisdn` is 15 digits.

##### **IOWA\_COAP\_403\_FORBIDDEN**

`shortID` is either zero or **IOWA\_LWM2M\_ID\_ALL**.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`shortID` does not match any known server.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`iowa_client.h`

#### Notes

`iowa_client_set_server_msisdn()` can only be called when IOWA is built with the flag **IOWA\_SMS\_SUPPORT**.

To unset the MSISDN, the parameter `msisdn` can take the value NULL.

An MSISDN can not be set for the Bootstrap Server.



### 3.4.10 iowa\_client\_set\_server\_registration\_behaviour

#### Prototype

```
iowa_status_t iowa_client_set_server_registration_behaviour(iowa_context_t contextP,
                                                           uint16_t shortId,
                                                           uint16_t priorityOrder,
                                                           int32_t initialDelayTimer,
                                                           bool blockOnFailure,
                                                           bool bootstrapOnFailure);
```

#### Description

`iowa_client_set_server_registration_behaviour()` set the registration behaviour of a LwM2M Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **shortID**

The ID assigned to the server.

##### **priorityOrder**

The Server priority order for the registration sequence.

##### **initialDelayTimer**

The initial delay to wait before to send the registration.

##### **blockOnFailure**

If registration fails and true is set, the registration sequence is interrupted.

##### **bootstrapOnFailure**

If registration fails and true is set, a bootstrap sequence is initiated.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- *initialDelayTimer* is negative.
- *bootstrapOnFailure* is equals to true but `[LWM2M_BOOTSTRAP][LWM2M_BOOTSTRAP]` is not set.

##### **IOWA\_COAP\_403\_FORBIDDEN**

*shortID* is either zero or `IOWA_LWM2M_ID_ALL`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

*shortID* does not match any known server.

#### Header File

`iowa_client.h`

#### Notes

This API requires `LWM2M_VERSION_1_1_SUPPORT` to be set.

If `IOWA_SERVER_RSC_REGISTRATION_BEHAVIOUR_REMOVE` is set, this API cannot be called.

### 3.4.11 iowa\_client\_set\_server\_communication\_attempts

#### Prototype

```
iowa_status_t iowa_client_set_server_communication_attempts(iowa_context_t contextP,
                                                           uint16_t shortId,
                                                           uint8_t retryCount,
                                                           int32_t retryDelayTimer,
                                                           uint8_t sequenceRetryCount,
                                                           int32_t sequenceDelayTimer);
```

#### Description

`iowa_client_set_server_communication_attempts()` set the communication attempts of a LwM2M Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **shortID**

The ID assigned to the server.

##### **retryCount**

The number of successive registration attempts before which a registration sequence is considered as failed.

##### **retryDelayTimer**

The number to wait between each registration sequence. The value is multiplied by two to the power of the registration retry attempt minus one ( $2^{**}(\text{retry attempt}-1)$ ) to create an exponential back-off.

##### **sequenceRetryCount**

The number of successive registration sequences before which a registration attempt is considered as failed.

##### **sequenceDelayTimer**

The number to wait between each successive registration sequences. The value is multiplied by two to the power of the registration retry attempt minus one ( $2^{**}(\text{retry attempt}-1)$ ) to create an exponential back-off.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- *retryCount* is superior to 32.
- *retryDelayTimer* is negative.
- *sequenceRetryCount* is superior to 32.
- *sequenceDelayTimer* is negative.

##### **IOWA\_COAP\_403\_FORBIDDEN**

*shortID* is either zero or **IOWA\_LWM2M\_ID\_ALL**.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

*shortID* does not match any known server.

#### Header File

`iowa_client.h`

## Notes

This API requires **LWM2M\_VERSION\_1\_1\_SUPPORT** to be set.

If **IOWA\_SERVER\_RSC\_COMMUNICATION\_ATTEMPTS\_REMOVE** is set, this API cannot be called.

Evaluation

### 3.4.12 iowa\_client\_get\_server\_coap\_peer

#### Prototype

```
iowa_coap_peer_t * iowa_client_get_server_coap_peer(iowa_context_t contextP,  
                                                    uint16_t shortId);
```

#### Description

`iowa_client_get_server_coap_peer()` returns the CoAP peer associated to a LwM2M Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **shortID**

The ID assigned to the server.

#### Return Value

A pointer to the `iowa_coap_peer_t` associated to the LwM2M Server.

This pointer may be nil if `shortId` is invalid or if IOWA did not yet initiate the registration to the LwM2M Server.

#### Header File

`iowa_client.h`

### 3.4.13 iowa\_client\_set\_notification\_default\_periods

#### Prototype

```
iowa_status_t iowa_client_set_notification_default_periods(iowa_context_t contextP,
                                                         uint16_t shortID,
                                                         uint32_t minPeriod,
                                                         uint32_t maxPeriod);
```

#### Description

`iowa_client_set_notification_default_periods()` configures the default periods for notifications sent to a LwM2M Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **shortID**

The ID assigned to the server or `IOWA_LWM2M_ID_ALL`.

##### **minPeriod**

The default minimum time in seconds between two notifications sent to the LwM2M Server for the same observation.

##### **maxPeriod**

The default maximum time in seconds between two notifications sent to the LwM2M Server for the same observation.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_403\_FORBIDDEN**

`shortID` is zero.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`shortID` does not match any known server.

#### Header File

`iowa_client.h`

#### Notes

When `IOWA_LWM2M_ID_ALL` is used as `shortID`, only already known LwM2M Servers will have the default periods configured. If a LwM2M Server is added after the call to this API, by default it will not have default periods.

Setting the default periods does not affect already running observations.

A minimum period set to zero is equivalent to having no minimum period defined. Same for maximum period.

If `maxPeriod` is inferior to `minPeriod`, it is cleared (i.e. set to zero).

### 3.4.14 iowa\_client\_use\_reliable\_notifications

#### Prototype

```
iowa_status_t iowa_client_use_reliable_notifications(iowa_context_t contextP,  
                                                    uint16_t shortId,  
                                                    bool enable);
```

#### Description

`iowa_client_use_reliable_notifications()` configures the LwM2M Client to ensure that notifications are received by the LwM2M Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **shortId**

The ID assigned to the server or `IOWA_LWM2M_ID_ALL`.

##### **enable**

If true, notifications will be reliable.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_403\_FORBIDDEN**

`shortID` is zero.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`shortID` does not match any known server.

#### Header File

`iowa_client.h`

#### Notes

When `IOWA_LWM2M_ID_ALL` is used as `shortID`, only already known LwM2M Servers will have reliable notifications. If a LwM2M Server is added after the call to this API, by default it will not use reliable notifications.

If `enable` is true:

- on unreliable transports like UDP, the notifications are sent as Confirmable messages.
- if a notification does not reach the LwM2M Server, IOWA stores it until the LwM2M Server is reachable again. See [\[LWM2M\\_STORAGE\\_QUEUE\\_SUPPORT\]](#)`[LWM2M_STORAGE_QUEUE_SUPPORT]` and [LWM2M\\_STORAGE\\_QUEUE\\_PEEK\\_SUPPORT](#).

### 3.4.15 iowa\_client\_object\_set\_mode

#### Prototype

```
iowa_status_t iowa_client_object_set_mode(iowa_context_t contextP,  
                                          iowa_sensor_t id,  
                                          uint8_t mode);
```

#### Description

`iowa_client_object_set_mode()` sets the sensor mode.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

The ID of the sensor.

##### **mode**

Flags used to enable modes.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`id` does not match any known sensor.

#### Header File

`iowa_client.h`

#### Notes

To use this API, the compilation flag `[LWM2M_CLIENT_ASYNCHRONOUS_OPERATION_SUPPORT][LWM2M_CLIENT_ASYNCHRONOUS_OPERATION_SUPPORT]` must be set.

To set the sensor mode, you can use the following flag:

- `IOWA_OBJECT_MODE_DEFAULT`
- `IOWA_OBJECT_MODE_ASYNCHRONOUS`

By default, sensors are synchronous.

A call to `iowa_client_object_set_mode()` affects all the sensors of the same type.

### 3.4.16 iowa\_client\_device\_update\_battery

#### Prototype

```
iowa_status_t iowa_client_device_update_battery(iowa_context_t contextP,
                                                uint8_t batteryLevel,
                                                iowa_device_battery_status_t batteryStatus
                                                );
```

#### Description

`iowa_client_device_update_battery()` updates the battery level and status exposed in the [Device Object][Device Object].

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **batteryLevel**

The battery level in percent.

##### **batteryStatus**

The battery status.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**

client has been configured without the flag **IOWA\_DEVICE\_RSC\_BATTERY** in the `iowa_device_info_t` structure. To reconfigure the client, close then reopen a fresh IOWA Client context with `iowa_close()`, `iowa_init()` and `iowa_client_configure()`.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

`batteryLevel` is outside the range [0; 100].

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

client is not configured. Call first `iowa_client_configure()`.

#### Header File

`iowa_client.h`

#### Notes

For the device to expose its battery level and status, `iowa_client_configure()` must have been called with the **IOWA\_DEVICE\_RSC\_BATTERY** flag.

Before the first call to `iowa_client_device_update_battery()`, default value of `batteryStatus` is **IOWA\_DEVICE\_BATTERY\_STATUS\_UNKNOWN**.

`iowa_client_device_update_battery()` can only be called when IOWA is built WITHOUT the flag **IOWA\_DEVICE\_RSC\_BATTERY\_REMOVE**.

#### **iowa\_device\_battery\_status\_t**

This is an enumeration of the following values:



**IOWA\_DEVICE\_BATTERY\_STATUS\_NORMAL**

The battery is operating normally and not on power.

**IOWA\_DEVICE\_BATTERY\_STATUS\_CHARGING**

The battery is currently charging.

**IOWA\_DEVICE\_BATTERY\_STATUS\_CHARGE\_COMPLETE**

The battery is fully charged and still on power.

**IOWA\_DEVICE\_BATTERY\_STATUS\_DAMAGED**

The battery has some problem.

**IOWA\_DEVICE\_BATTERY\_STATUS\_LOW\_BATTERY**

The battery is low on charge.

**IOWA\_DEVICE\_BATTERY\_STATUS\_NOT\_INSTALLED**

The battery is not installed.

**IOWA\_DEVICE\_BATTERY\_STATUS\_UNKNOWN**

The battery information is not available.

Evaluation

### 3.4.17 iowa\_client\_add\_device\_power\_source

#### Prototype

```
iowa_status_t iowa_client_add_device_power_source(iowa_context_t context,
                                                  iowa_power_source_type_t type,
                                                  int voltageValue,
                                                  int currentValue,
                                                  iowa_sensor_t *idP);
```

#### Description

`iowa_client_add_device_power_source()` adds a power source to Device object with initial value of voltage and current.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **type**

power source type.

##### **voltageValue**

initial voltage value (mV).

##### **currentValue**

initial current value (mA).

##### **idP**

Used to store the ID of the created power source. Not checked at runtime.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

client is not configured. Call first `iowa_client_configure()`.

##### **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**

client has been configured without the flag **IOWA\_DEVICE\_RSC\_POWER\_SOURCE** in the `iowa_device_info_t` structure. To reconfigure the client, close then reopen a fresh IOWA Client context with `iowa_close()`, `iowa_init()` and `iowa_client_configure()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`iowa_client.h`

#### Notes

For the device to expose its power source information, `iowa_client_configure()` must have been called with the **IOWA\_DEVICE\_RSC\_POWER\_SOURCE** flag.

To update a power source values, you need to call `iowa_client_update_device_power_source()`.

To remove a power source, you need to call `iowa_client_remove_device_power_source()`.

`iowa_client_add_device_power_source()` can only be called when IOWA is built WITHOUT the flag **IOWA\_DEVICE\_RSC\_POWER\_SOURCE\_REMOVE**.

## **iowa\_power\_source\_type\_t**

This is an enumeration of the following values:

### **IOWA\_POWER\_SOURCE\_DC\_POWER**

DC power supply.

### **IOWA\_POWER\_SOURCE\_INTERNAL\_BATTERY**

Internal battery.

### **IOWA\_POWER\_SOURCE\_EXTERNAL\_BATTERY**

External battery.

### **IOWA\_POWER\_SOURCE\_FUEL\_CELL**

Fuel Cell

### **IOWA\_POWER\_SOURCE\_POWER\_OVER\_ETHERNET**

Power Over Ethernet.

### **IOWA\_POWER\_SOURCE\_USB**

USB.

### **IOWA\_POWER\_SOURCE\_AC\_MAIN\_POWER**

AC power supply.

### **IOWA\_POWER\_SOURCE\_SOLAR**

Solar energy.

### 3.4.18 iowa\_client\_remove\_device\_power\_source

#### Prototype

```
iowa_status_t iowa_client_remove_device_power_source(iowa_context_t context,
                                                    iowa_sensor_t id);
```

#### Description

`iowa_client_remove_device_power_source()` removes a power source from the Device object.

#### Arguments

##### *contextP*

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### *id*

ID of the power source to remove.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

client is not configured. Call first `iowa_client_configure()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

*id* is not a device's power source. Valid *id* are only returned by `iowa_client_add_device_power_source()`.

##### **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**

client has been configured without the flag **IOWA\_DEVICE\_RSC\_POWER\_SOURCE** in the `iowa_device_info_t` structure. To reconfigure the client, close then reopen a fresh IOWA Client context with `iowa_close()`, `iowa_init()` and `iowa_client_configure()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`iowa_client.h`

#### Notes

For the device to expose its power source information, `iowa_client_configure()` must have been called with the **IOWA\_DEVICE\_RSC\_POWER\_SOURCE** flag.

`iowa_client_remove_device_power_source()` can only be called when IOWA is built WITHOUT the flag **IOWA\_DEVICE\_RSC\_POWER\_SOURCE\_REMOVE**.

### 3.4.19 iowa\_client\_update\_device\_power\_source

#### Prototype

```
iowa_status_t iowa_client_update_device_power_source(iowa_context_t context,
                                                    iowa_sensor_t id,
                                                    int voltageValue,
                                                    int currentValue);
```

#### Description

`iowa_client_update_device_power_source()` updates a power source values to Device object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the power source.

##### **voltageValue**

new voltage value (mV).

##### **currentValue**

new current value (mA).

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

client is not configured. Call first `iowa_client_configure()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`id` is not a device's power source. Valid `id` are only returned by `iowa_client_add_device_power_source()`.

##### **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**

client has been configured without the flag **IOWA\_DEVICE\_RSC\_POWER\_SOURCE** in the `iowa_device_info_t` structure. To reconfigure the client, close then reopen a fresh IOWA Client context with `iowa_close()`, `iowa_init()` and `iowa_client_configure()`.

#### Header File

`iowa_client.h`

#### Notes

For the device to expose its power source information, `iowa_client_configure()` must have been called with the **IOWA\_DEVICE\_RSC\_POWER\_SOURCE** flag.

`iowa_client_update_device_power_source()` can only be called when IOWA is built WITHOUT the flag **IOWA\_DEVICE\_RSC\_POWER\_SOURCE\_REMOVE**.

### 3.4.20 iowa\_client\_set\_device\_error\_code

#### Prototype

```
iowa_status_t iowa_client_set_device_error_code(iowa_context_t context,
                                                uint8_t errorCode);
```

#### Description

`iowa_client_set_device_error_code()` sets an error code on Device object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **errorCode**

The error code value to set.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

client is not configured. Call first `iowa_client_configure()`.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`errorCode` is not a valid parameter.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`errorCode` is `IOWA_ERROR_CODE_NO_ERROR` but there is no error to clear.

##### **IOWA\_COAP\_409\_CONFLICT**

`errorCode` has already been set.

#### Header File

`iowa_client.h`

#### Notes

To clear one error code, you need to call `iowa_client_clear_device_error_code()`.

To clear all error codes, you can call `iowa_client_set_device_error_code()` with `errorCode` argument set to `IOWA_ERROR_CODE_NO_ERROR`.

Error code values are:

- **IOWA\_ERROR\_CODE\_NO\_ERROR**  
No error.
- **IOWA\_ERROR\_CODE\_LOW\_BATTERY\_POWER**  
Low battery power.
- **IOWA\_ERROR\_CODE\_EXTERNAL\_POWER\_SUPPLY\_OFF**  
External power supply off.
- **IOWA\_ERROR\_CODE\_GPS\_MODULE\_FAILURE**  
GPS module failure.
- **IOWA\_ERROR\_CODE\_LOW\_RECEIVED\_SIGNAL\_STRENGTH**  
Low received signal strength.

- **IOWA\_ERROR\_CODE\_OUT\_OF\_MEMORY**  
Out of memory.
- **IOWA\_ERROR\_CODE\_SMS\_FAILURE**  
SMS failure.
- **IOWA\_ERROR\_CODE\_IP\_CONNECTIVITY\_FAILURE**  
IP connectivity failure.
- **IOWA\_ERROR\_CODE\_PERIPHERAL\_MALFUNCTION**  
Peripheral malfunction.

Evaluation

### 3.4.21 iowa\_client\_clear\_device\_error\_code

#### Prototype

```
iowa_status_t iowa_client_clear_device_error_code(iowa_context_t context,
                                                  uint8_t errorCode);
```

#### Description

`iowa_client_clear_device_error_code()` clears an error code from the Device object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **errorCode**

The error code to clear. It can't be `IOWA_ERROR_CODE_NO_ERROR`.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

client is not configured. Call first `iowa_client_configure()`.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

The error code `IOWA_ERROR_CODE_NO_ERROR` can't be cleared.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

The error code is not set.

#### Header File

`iowa_client.h`

#### Notes

Error code values are enumerated in `iowa_client_set_device_error_code()`.



### 3.4.22 iowa\_client\_update\_device\_time\_information

#### Prototype

```
iowa_status_t iowa_client_update_device_time_information(iowa_context_t contextP,  
                                                         iowa_device_time_info_t *  
                                                         timeInfoP);
```

#### Description

`iowa_client_update_device_time_information()` updates time information to Device object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **timeInfoP**

Current device time information: `iowa_device_time_info_t`.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

client is not configured. Call first `iowa_client_configure()`.

#### Header File

`iowa_client.h`

#### Notes

For the device to expose its time information, `iowa_client_configure()` must have been called with time information used.

### 3.4.23 iowa\_client\_add\_custom\_object

#### Prototype

```
iowa_status_t iowa_client_add_custom_object(iowa_context_t contextP,
                                            uint16_t objectID,
                                            size_t instanceCount,
                                            uint16_t * instanceIDs,
                                            size_t resourceCount,
                                            iowa_lwm2m_resource_desc_t * resourceArray,
                                            iowa_RWE_callback_t dataCallback,
                                            iowa_CD_callback_t instanceCallback,
                                            iowa_RI_callback_t resInstanceCallback,
                                            void * userData);
```

#### Description

`iowa_client_add_custom_object()` adds a new custom Object for the LwM2M Client to handle. The object is defined by its ID and a the list of the resources it contains.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **objectID**

The ID of the Object.

##### **instanceCount**

The number of elements in `instanceIDs`. This can be 0.

##### **instanceIDs**

The IDs of the instances of the Object. This can be nil.

##### **resourceCount**

The number of elements in `resourceArray`.

##### **resourceArray**

An array of `iowa_lwm2m_resource_desc_t` composing the Object.

##### **dataCallback**

The callback to perform Read, Write and Execute operations on the resources.

##### **instanceCallback**

The callback to perform Create and Delete operations on Object instances. This can be nil.

##### **resInstanceCallback**

The callback to retrieve the list of instances of resources declared as multiple. This can be nil.

##### **userData**

Passed as argument to the callbacks.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_403\_FORBIDDEN**

objectID is 0, 1 or 3 which are reserved Object IDs.

### IOWA\_COAP\_406\_NOT\_ACCEPTABLE

either:

- *objectID* is *IOWA\_LWM2M\_ID\_ALL* (65535).
- *resourceCount* is zero.
- *resourceArray* is nil.
- *dataCallback* is nil.
- *instanceIDs* is nil and *instanceCount* is not zero.
- *resInstanceCallback* is nil and one of the resources in *resourceArray* has the **IOWA\_RESOURCE\_FLAG\_MULTIPLE** flag set.

### IOWA\_COAP\_409\_CONFLICT

this object already exists. Call first `iowa_client_remove_custom_object()`.

#### Header File

`iowa_client.h`

#### Notes

Object IDs 0, 1 and 3 are reserved and cannot be used.

Per Lightweight M2M specification, the ID of the instance of a single-instance Object is 0. When creating a single-instance Object, you can set *instanceCount* to zero and *instanceCallback* to nil. IOWA will automatically create an instance with ID 0.

When the LwM2M Server creates a new instance of the custom object, *instanceCallback* is first called with the new instance ID then *dataCallback* is called with *operation* set to **IOWA\_DM\_WRITE** to initialize the instance. Thus if *instanceCallback* is defined, *dataCallback* must handle the Write operation even on resources declared as read-only.

### 3.4.24 iowa\_client\_remove\_custom\_object

#### Prototype

```
iowa_status_t iowa_client_remove_custom_object(iowa_context_t contextP,  
                                              uint16_t objectID);
```

#### Description

`iowa_client_remove_custom_object()` removes a custom Object created with `iowa_client_add_custom_object()`.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **objectID**

The ID of the Object.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_403\_FORBIDDEN**

*objectID* is 0, 1 or 3 which are reserved Object IDs.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

*objectID* does not match any known object.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

*objectID* is **IOWA\_LWM2M\_ID\_ALL** (65535).

#### Header File

`iowa_client.h`

### 3.4.25 iowa\_client\_object\_resource\_changed

#### Prototype

```
iowa_status_t iowa_client_object_resource_changed(iowa_context_t contextP,  
                                                  uint16_t objectID,  
                                                  uint16_t instanceID,  
                                                  uint16_t resourceID);
```

#### Description

`iowa_client_object_resource_changed()` informs the IOWA stack that the value of a LwM2M Object resource changed.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **objectID**

The ID of the Object containing the resource.

##### **instanceID**

The ID of the Instance containing the resource. This can be **IOWA\_LWM2M\_ID\_ALL**.

##### **resourceID**

The ID of the resource. This can be **IOWA\_LWM2M\_ID\_ALL**.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_403\_FORBIDDEN**

`objectID` is 0, 1 or 3 which are reserved Object IDs.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

`objectID` is **IOWA\_LWM2M\_ID\_ALL** (65535).

#### Header File

`iowa_client.h`

#### Notes

This API does not check if the LwM2M Object resource exists. That's why this API does not return **IOWA\_COAP\_404\_NOT\_FOUND**. Actually, `iowa_client_object_resource_changed()` is only searching a match between the running observation and the URI provided. If a match is found a notification is sent, else nothing happens.

### 3.4.26 iowa\_client\_object\_instance\_changed

#### Prototype

```
iowa_status_t iowa_client_object_instance_changed(iowa_context_t contextP,  
                                                  uint16_t objectID,  
                                                  uint16_t instanceID,  
                                                  iowa_dm_operation_t operation);
```

#### Description

`iowa_client_object_instance_changed()` informs the IOWA stack that an instance of a LwM2M Object was created or deleted.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **objectID**

The ID of the Object containing the instance.

##### **instanceID**

The ID of the created or deleted Instance.

##### **operation**

**IOWA\_DM\_CREATE** if it is a new instance. **IOWA\_DM\_DELETE** if the instance was removed.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_403\_FORBIDDEN**

*objectID* is 0, 1 or 3 which are reserved Object IDs.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

either:

- *objectID* does not match any known object.
- *operation* is **IOWA\_DM\_DELETE** and *instanceID* does not match any known instance.

##### **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**

*operation* is neither **IOWA\_DM\_CREATE** nor **IOWA\_DM\_DELETE**.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- *objectID* is **IOWA\_LWM2M\_ID\_ALL** (65535).
- *operation* is **IOWA\_DM\_CREATE** and *instanceID* was already present.

#### Header File

`iowa_client.h`

### 3.4.27 iowa\_client\_notification\_lock

#### Prototype

```
void iowa_client_notification_lock(iowa_context_t contextP,  
                                  bool enter);
```

#### Description

`iowa_client_notification_lock()` prevents or allows the IOWA stack to send notifications.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **enter**

**true** to stop the notification, **false** to resume the notification.

#### Return Value

None.

#### Header File

`iowa_client.h`

#### Notes

The main use is to perform several calls to `iowa_client_object_resource_changed()` on an Object without generating a notification each time if the Object is under observation.

This function is useful only if IOWA is built with the **IOWA\_MULTITHREAD\_SUPPORT** flag. Inside a custom object callback, notifications are already disabled.

### 3.4.28 iowa\_client\_send\_heartbeat

#### Prototype

```
iowa_status_t iowa_client_send_heartbeat(iowa_context_t contextP,  
                                         uint16_t shortID);
```

#### Description

`iowa_client_send_heartbeat()` sends an heartbeat message to a server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **shortID**

The Short ID assigned to this Server. Can be equal to **IOWA\_LWM2M\_ID\_ALL**.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_403\_FORBIDDEN**

*shortID* is zero.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

*shortID* does not match any known server.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

client is not connected to the server with *shortID*. This can happen when:

- The Server is a Bootstrap Server and the Client is already connect to a Server.
- The Client is configured with more than one Server and has established the connection with only one.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

heartbeat message has not been sent by the platform.

#### Header File

`iowa_client.h`

#### Notes

If *shortID* is equal to **IOWA\_LWM2M\_ID\_ALL**, the heartbeat message will be sent to all servers.

For non LoRaWAN Servers, a registration update message is sent to the Server. The `iowa_event_callback_t` will be called with a **IOWA\_EVENT\_REG\_UPDATING** event. Then, if a reply is received from the Server, the `iowa_event_callback_t` will be called with either a **IOWA\_EVENT\_REG\_REGISTERED** or **IOWA\_EVENT\_REG\_FAILED** event. Nothing is done when no reply is received from the Server.

In the **IOWA\_EVENT\_REG\_REGISTERED** case, the registration lifetime timer for the LwM2M Server is resetted.



### 3.4.29 iowa\_client\_send\_sensor\_data

#### Prototype

```
iowa_status_t iowa_client_send_sensor_data(iowa_context_t contextP,
                                           uint16_t shortId,
                                           iowa_sensor_uri_t *sensorUriP,
                                           size_t sensorUriCount,
                                           iowa_response_callback_t responseCb,
                                           void *userDataP);
```

#### Description

`iowa_client_send_sensor_data()` sends data from `iowa_sensor_t` to server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **shortId**

The ID of the server. It can be `IOWA_LWM2M_ID_ALL` to send to all registered servers.

##### **sensorUriP, sensorUriCount**

The sensor uri to send.

##### **responseCb**

The callback called when the reply to this operation is known. This can be nil.

##### **userDataP**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

`sensorUriCount` is zero or `sensorUriP` is nil.

##### **IOWA\_COAP\_401\_UNAUTHORIZED**

The destination LwM2M Server does not have the Read Access Right to the sent data. Refer to the [Access Control List Object][Access Control List Object] for details.

##### **IOWA\_COAP\_403\_FORBIDDEN**

`shortId` is not an acceptable value.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

either:

- `shortId` does not match a known server.
- at least one `sensorUriP[x]` does not match a known resource.

##### **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**

at least one `sensorUriP[x]`'s resource is not readable.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

the receiving LwM2M Server has muted the Send feature. See the Mute Send resource of the [Server Object][Server Object].

##### **IOWA\_COAP\_413\_REQUEST\_ENTITY\_TOO\_LARGE**

The Platform abstraction didn't send all the data. One possible assumption is the packet was too large for the transport.

**IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

the client is not registered to the requested LwM2M Server or the communication with the requested LwM2M Server failed.

**Header File**

iowa\_client.h

**Notes**

This API requires the compilation flag `[LWM2M_DATA_PUSH_SUPPORT][LWM2M_DATA_PUSH_SUPPORT]`.

The *responseCb* will be called with the operation set to **IOWA\_DM\_DATA\_PUSH**.

If *shortId* is **IOWA\_LWM2M\_ID\_ALL**, the *responseCb* will be called for each registered LwM2M Server which has not muted the Client.

Evaluation

### 3.4.30 iowa\_client\_send\_data

#### Prototype

```
iowa_status_t iowa_client_send_data(iowa_context_t contextP,
                                     uint16_t shortId,
                                     iowa_lwm2m_data_t *dataArrayP,
                                     size_t dataCount,
                                     iowa_response_callback_t responseCb,
                                     void *userDataP);
```

#### Description

`iowa_client_send_data()` sends data to server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **shortId**

The ID of the server. It can be `IOWA_LWM2M_ID_ALL` to send to all registered servers.

##### **dataArrayP, dataCount**

The data to send.

##### **responseCb**

The callback called when the reply to this operation is known. This can be nil.

##### **resultUserDataP**

A pointer to application specific data. This is passed as argument to `responseCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- at least one `dataArrayP[x].resourceID` is `IOWA_LWM2M_ID_ALL`.
- `dataCount` is zero or `dataArrayP` is nil.

##### **IOWA\_COAP\_401\_UNAUTHORIZED**

The destination Lwm2m Server does not have the Read Access Right to the sent data. Refer to the [Access Control List Object][Access Control List Object] for details.

##### **IOWA\_COAP\_403\_FORBIDDEN**

`shortId` is not an acceptable value.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

either:

- `shortId` does not match a known server.
- at least one `dataArrayP[x]` does not match a known resource.

##### **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**

at least one `dataArrayP[x]`'s resource is not readable.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- one of the timestamped value has an negative timestamp.
- at least one *dataArrayP[x]* has negative value with unsigned integer type

**IOWA\_COAP\_412\_PRECONDITION\_FAILED**

the receiving LwM2M Server has muted the Send feature. See the Mute Send resource of the [Server Object][Server Object].

**IOWA\_COAP\_413\_REQUEST\_ENTITY\_TOO\_LARGE**

The Platform abstraction didn't send all the data. One possible assumption is the packet was too large for the transport.

**IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

the client is not registered to the requested LwM2M Server or the communication with the requested LwM2M Server failed.

**Header File**

iowa\_client.h

**Notes**

This API requires the compilation flag **[LWM2M\_DATA\_PUSH\_SUPPORT][LWM2M\_DATA\_PUSH\_SUPPORT]**.

The *responseCb* will be called with the operation set to **IOWA\_DM\_DATA\_PUSH**.

If *shortId* is **IOWA\_LWM2M\_ID\_ALL**, the *responseCb* will be called for each registered LwM2M Server which has not muted the Client.

## 3.5 Accelerometer Object API

This IPSO object can be used to represent a 1-3 axis accelerometer.

To be able to use this object, `iowa_accelerometer.h` must be included.

### 3.5.1 iowa\_client\_add\_accelerometer\_object

#### Prototype

```
iowa_status_t iowa_client_add_accelerometer_object(iowa_context_t context,
                                                    uint16_t optFlags,
                                                    float minRangeValue,
                                                    float maxRangeValue,
                                                    const char *sensorUnits,
                                                    iowa_sensor_t *idP);
```

#### Description

`iowa_client_add_accelerometer_object()` creates an accelerometer object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to add optional resources.

##### **minRangeValue**

Minimal range value for the accelerometer.

##### **maxRangeValue**

Maximal range value for the accelerometer.

##### **sensorUnits**

Measurement units definition

##### **idP**

Used to store the ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

*minRangeValue* argument is superior to *maxRangeValue* argument.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`objects/iowa_accelerometer.h`

#### Notes

Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- IOWA\_ACCELEROMETER\_RSC\_Y\_VALUE
- IOWA\_ACCELEROMETER\_RSC\_Z\_VALUE
- IOWA\_ACCELEROMETER\_RSC\_MIN\_RANGE\_VALUE
- IOWA\_ACCELEROMETER\_RSC\_MAX\_RANGE\_VALUE

Moreover, you can add several optional resources at one time by using the following flags:

- IOWA\_ACCELEROMETER\_3\_AXIS
- IOWA\_ACCELEROMETER\_RANGE\_VALUE

Evaluation

### 3.5.2 iowa\_client\_remove\_accelerometer\_object

#### Prototype

```
iowa_status_t iowa_client_remove_accelerometer_object(iowa_context_t context,
                                                    iowa_sensor_t id);
```

#### Description

`iowa_client_remove_accelerometer_object()` removes an accelerometer object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not an accelerometer object. Valid `id` are only returned by `iowa_client_add_accelerometer_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

accelerometer referred by `id` does not exist.

#### Header File

`objects/iowa_accelerometer.h`

### 3.5.3 iowa\_client\_accelerometer\_update\_axis

#### Prototype

```
iowa_status_t iowa_client_accelerometer_update_axis(iowa_context_t context,
                                                    iowa_sensor_t id,
                                                    float xValue,
                                                    float yValue,
                                                    float zValue);
```

#### Description

`iowa_client_accelerometer_update_axis()` updates values of an accelerometer object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

##### **xValue**

X value axis

##### **yValue**

Y value axis

##### **zValue**

Z value axis

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not an accelerometer object. Valid `id` are only returned by `iowa_client_add_accelerometer_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

accelerometer referred by `id` does not exist.

#### Header File

`objects/iowa_accelerometer.h`



## 3.6 Access Control List Object API

This LwM2M Object is used to check whether the LwM2M Server has access right for performing an operation.

To be able to use this object, `iowa_access_control_list.h` must be included and the define `[IOWA_SUPPORT_ACCESS_CONTROL_LIST_OBJECT][IOWA_SUPPORT_ACCESS_CONTROL_LIST_OBJECT]` must be set.

### 3.6.1 iowa\_client\_acl\_rights\_server\_set

#### Prototype

```
iowa_status_t iowa_client_acl_rights_server_set(iowa_context_t contextP,
                                                uint16_t objectId,
                                                uint16_t instanceId,
                                                uint16_t serverId,
                                                uint8_t accessRights);
```

#### Description

`iowa_client_acl_rights_server_set()` set the access rights for a LwM2M Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **objectId**

ID of the Object.

##### **instanceId**

ID of the Object Instance.

##### **serverId**

Short Server ID of a LwM2M Server or `IOWA_ACL_DEFAULT_ID`.

##### **accessRights**

new access rights to set.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

either:

if `instanceId` is `IOWA_LWM2M_ID_ALL`, `accessRights` must be `IOWA_ACL_CREATE_RIGHT`.

if `instanceId` is not `IOWA_LWM2M_ID_ALL`, `accessRights` cannot include `IOWA_ACL_CREATE_RIGHT`.

##### **IOWA\_COAP\_403\_FORBIDDEN**

either:

`objectId` is `IOWA_LWM2M_ID_ALL`.

`serverId` is `IOWA_LWM2M_ID_ALL`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

either:

`objectId` does not refer to a supported Object.

if `instanceId` is not `IOWA_LWM2M_ID_ALL`, `instanceId` does not refer to an instantiated Object Instance.

`serverId` does not refer to a known Server Short ID nor `IOWA_ACL_DEFAULT_ID`.

## IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR

a memory allocation failed.

### Header File

objects/iowa\_access\_control\_list.h

### Notes

*accessRights* is a bit field which can contain the following values:

- *IOWA\_ACL\_NONE\_RIGHT*: No access
- *IOWA\_ACL\_READ\_RIGHT*: Read access
- *IOWA\_ACL\_WRITE\_RIGHT*: Write access
- *IOWA\_ACL\_EXECUTE\_RIGHT*: Execute access
- *IOWA\_ACL\_DELETE\_RIGHT*: Delete access
- *IOWA\_ACL\_CREATE\_RIGHT*: Create access

If access rights are already set for the targeted *objectId*, *instanceId* and *serverId*, they will be overwritten.

Access rights set through *iowa\_client\_acl\_rights\_server\_set()* cannot be modified by any Server, since the Server Owner ID will be **IOWA\_LWM2M\_ID\_ALL** (means Bootstrap Server).

To set the default access rights, *serverId* can be **IOWA\_ACL\_DEFAULT\_ID**.

### 3.6.2 iowa\_client\_acl\_rights\_server\_clear

#### Prototype

```
iowa_status_t iowa_client_acl_rights_server_clear(iowa_context_t contextP,
                                                uint16_t objectId,
                                                uint16_t instanceId,
                                                uint16_t serverId);
```

#### Description

`iowa_client_acl_rights_server_clear()` unset the access rights for a LwM2M Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **objectId**

ID of the Object.

##### **instanceId**

ID of the Object Instance.

##### **serverId**

Short Server ID of a LwM2M Server or **IOWA\_ACL\_DEFAULT\_ID**.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_403\_FORBIDDEN**

*objectId* or *serverId* is **IOWA\_LWM2M\_ID\_ALL**.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

*objectId*, *instanceId* or *serverId* do not have access rights set.

#### Header File

objects/iowa\_access\_control\_list.h

### 3.6.3 iowa\_client\_acl\_rights\_object\_clear

#### Prototype

```
iowa_status_t iowa_client_acl_rights_object_clear(iowa_context_t contextP,  
                                                uint16_t objectId,  
                                                uint16_t instanceId);
```

#### Description

`iowa_client_acl_rights_object_clear()` clears the access rights for an Object/Object Instance.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **objectId**

ID of the Object.

##### **instanceId**

ID of the Object Instance.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_403\_FORBIDDEN**

`objectId` is **IOWA\_LWM2M\_ID\_ALL**.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`objectId` or `instanceId` do not have access rights set.

#### Header File

`objects/iowa_access_control_list.h`

## 3.7 APN Connection Profile Object API

This LwM2M object specifies resources to enable a device to connect to an APN.

To be able to use this object, `iowa_apn_connection_profile.h` must be included.

### 3.7.1 Data Structures and Constants

#### 3.7.1.1 iowa\_apn\_connection\_profile\_details\_t

```
typedef struct
{
    char        *apn;
    bool        autoSelect;
    bool        enableStatus;
    int         authenticationType;
    char        *userName;
    char        *secret;
    char        *reconnectSchedule;
    char        **validityList;
    uint16_t    validityNumber;
    int         *connectionEstablishmentTimeList;
    uint16_t    connectionEstablishmentTimeNumber;
    int         *connectionEstablishmentResultList;
    uint16_t    connectionEstablishmentResultNumber;
    int         *connectionEstablishmentRejectCauseList;
    uint16_t    connectionEstablishmentRejectCauseNumber;
    int         *connectionEndTimeList;
    uint16_t    connectionEndTimeNumber;
    int         totalBytesSent;
    int         totalBytesReceived;
    char        **ipAddressList;
    uint16_t    ipAddressNumber;
    char        **prefixLengthList;
    uint16_t    prefixLengthNumber;
    char        **subnetMaskList;
    uint16_t    subnetMaskNumber;
    char        **gatewayList;
    uint16_t    gatewayNumber;
    char        **primaryDnsAddressList;
    uint16_t    primaryDnsAddressNumber;
    char        **secondaryDnsAddressList;
    uint16_t    secondaryDnsAddressNumber;
    int         qci;
    int         totalPacketsSent;
    int         pdnType;
    int         apnRateControl;
} iowa_apn_connection_profile_details_t;
```

#### ***apn***

APN of the APN connection profile.

#### ***autoSelect***

It enables the device to choose an APN according to a device specific algorithm.

#### ***enableStatus***

Allows the profile to be remotely activated or deactivated.

**authenticationType**

0: PAP, 1: CHAP, 2: PAP or CHAP, 3: None.

**userName**

Used with e.g. PAP.

**secret**

Used with e.g. PAP.

**reconnectSchedule**

List of retry delay values in seconds to be used in case of unsuccessful connection establishment attempts.

**validity**

Coma separated mobile country code, then mobile network code.

**connectionEstablishmentTime**

UTC time of connection request.

**connectionEstablishmentResult**

0 = accepted, 1 = rejected.

**connectionEstablishmentRejectCause**

Reject cause.

**connectionEndTime**

UTC time of connection end.

**totalBytesSent**

Rolling counter for total number of bytes sent via this interface since last device reset.

**totalBytesReceived**

Rolling counter for total number of bytes sent via this interface since last device reset.

**ipAddress**

May be IPv4 or IPv6 address.

**prefixLength**

Associated with IPv6 address.

**subnetMask**

Subnet mask.

**gateway**

Gateway.

**primaryDnsAddress**

Primary DNS address.

**secondaryDnsAddress**

Secondary DNS address.

**qci**

Quality of service Class Identifier. For LTE and NB-IoT only.

**totalPacketsSent**

Rolling counter for total number of packets sent via this interface since last device reset.

**pdnType**

0=Non-IP, 1=IPv4, 2=IPv6, 3=IPv4v6.

**apnRateControl**

Number of allowed uplink PDU transmissions per time interval per APN.

## 3.7.2 Callbacks

### 3.7.2.1 iowa\_apn\_connection\_profile\_update\_callback\_t

This callback is called when the Server writes new information on the APN connection profile object.

```
typedef iowa_status_t(*iowa_apn_connection_profile_update_callback_t)(
    char *profileName,
    iowa_dm_operation_t operation,
    uint32_t flags,
    iowa_apn_connection_profile_details_t *detailsP,
    void *userDataCallback,
    iowa_context_t contextP
);
```

**profileName**

Unique name of the APN connection profile. This may be new.

**operation**

The operation performed by the Server on this APN connection profile (creation, deletion, or write).

**flags**

Specify values set in detailsP.

**detailsP**

APN connection profile details. This may be nil.

**userDataCallback**

User data callback.

**contextP**

The IOWA context.

**Return Value** IOWA\_COAP\_NO\_ERROR in case of success or an error status.

### 3.7.3 API

#### 3.7.3.1 iowa\_client\_enable\_apn\_connection\_profile\_object

##### Prototype

```
iowa_status_t iowa_client_enable_apn_connection_profile_object(
    iowa_context_t contextP,
    iowa_apn_connection_profile_update_callback_t updateCallback,
    void *userDataCallback
);
```

**Description** `iowa_client_enable_apn_connection_profile_object()` enables APN connection profiles management.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **updateCallback**

Called to update state of the APN connection profile. This is called when the server request a new state.

###### **userDataCallback**

Application specific data pass to the callback. Can be nil.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

no update state callback provided means `updateCallback` is nil.

###### **IOWA\_COAP\_409\_CONFLICT**

APN connection profiles management was already enabled. Call first `iowa_client_disable_apn_connection_profile_object()`.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_apn_connection_profile.h`

**Notes** Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...



### 3.7.3.2 iowa\_client\_disable\_apn\_connection\_profile\_object

#### Prototype

```
iowa_status_t iowa_client_disable_apn_connection_profile_object(iowa_context_t contextP);
```

**Description** `iowa_client_disable_apn_connection_profile_object()` disables APN connection profiles management.

#### Arguments

##### *contextP*

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

APN connection profiles management was not enabled. `iowa_client_enable_apn_connection_profile_object()` was not called before, or failed.

**Header File** `objects/iowa_apn_connection_profile.h`

### 3.7.3.3 iowa\_client\_add\_apn\_connection\_profile

#### Prototype

```
iowa_status_t iowa_client_add_apn_connection_profile(iowa_context_t contextP,
                                                    const char *profileName,
                                                    uint32_t optFlags,
                                                    iowa_apn_connection_profile_details_t
                                                    *detailsP);
```

**Description** `iowa_client_add_apn_connection_profile()` add an APN connection profile.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **profileName**

Unique name of the APN connection profile.

##### **optFlags**

Optional flags to add optional resources.

##### **detailsP**

Apn connection profile details.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

inconsistent data inside `detailsP`.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- no profile name provided means `profileName` is nil.
- no details provided means `detailsP` is nil.

##### **IOWA\_COAP\_409\_CONFLICT**

APN connection profile with `profileName` already exists.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

APN connection profile management was not enabled. Call first `iowa_client_enable_apn_connection_profile_object()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_apn_connection_profile.h`

**Notes** When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_APN
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_AUTO\_SELECT\_APN\_DEVICE
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_ENABLE\_STATUS
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_USER\_NAME
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_SECRET

- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_RECONNECT\_SCHEDULE
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_VALIDITY
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_CONN\_ESTABLISHMENT\_TIME
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_CONN\_ESTABLISHMENT\_RESULT
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_CONN\_ESTABLISHMENT\_REJECT\_CAUSE
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_CONNECTION\_END\_TIME
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_TOTAL\_BYTES\_SENT
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_TOTAL\_BYTES\_RECEIVED
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_IP\_ADDRESS
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_PREFIX\_LENGTH
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_SUBNET\_MASK
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_GATEWAY
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_PRIMARY\_DNS\_ADDRESS
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_SECONDARY\_DNS\_ADDRESS
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_QCI
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_TOTAL\_PACKETS\_SENT
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_PDN\_TYPE
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_APN\_RATE\_CONTROL

Evaluation

### 3.7.3.4 iowa\_client\_remove\_apn\_connection\_profile

#### Prototype

```
iowa_status_t iowa_client_remove_apn_connection_profile(iowa_context_t contextP,  
                                                       const char *profileName);
```

**Description** `iowa_client_remove_apn_connection_profile()` removes an APN connection profile created with `iowa_client_add_apn_connection_profile()`.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **profileName**

Unique name of the APN connection profile.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

*profileName* does not match any known APN connection profile.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

no profile name provided meaning *profileName* is nil.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

APN connection profile management was not enabled. Call first `iowa_client_enable_apn_connection_profile_object()`.

**Header File** `objects/iowa_apn_connection_profile.h`

### 3.7.3.5 iowa\_client\_update\_apn\_connection\_profile

#### Prototype

```
iowa_status_t iowa_client_update_apn_connection_profile(
    iowa_context_t contextP,
    const char *profileName,
    uint32_t flags,
    iowa_apn_connection_profile_details_t *detailsP
);
```

**Description** `iowa_client_update_apn_connection_profile()` updates an APN connection profile.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **profileName**

Unique name of the APN connection profile.

##### **flags**

Specify resources to update.

##### **detailsP**

The APN connection profile details.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

inconsistent data inside *detailsP*.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

APN connection profile does not exist. Add first the profile with `iowa_client_add_apn_connection_profile()`.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- no profile name provided means *profileName* is nil.
- no details provided means *detailsP* is nil.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

APN connection profile management was not enabled. Call first `iowa_client_enable_apn_connection_profile_object()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_apn_connection_profile.h`

**Notes** To specify resources to update, you can use the following flags:

- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_APN
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_AUTO\_SELECT\_APN\_DEVICE
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_ENABLE\_STATUS
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_AUTHENTICATION\_TYPE
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_USER\_NAME

- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_SECRET
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_RECONNECT\_SCHEDULE
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_VALIDITY
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_CONN\_ESTABLISHMENT\_TIME
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_CONN\_ESTABLISHMENT\_RESULT
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_CONN\_ESTABLISHMENT\_REJECT\_CAUSE
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_CONNECTION\_END\_TIME
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_TOTAL\_BYTES\_SENT
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_TOTAL\_BYTES\_RECEIVED
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_IP\_ADDRESS
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_PREFIX\_LENGTH
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_SUBNET\_MASK
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_GATEWAY
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_PRIMARY\_DNS\_ADDRESS
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_SECONDARY\_DNS\_ADDRESS
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_QCI
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_TOTAL\_PACKETS\_SENT
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_PDN\_TYPE
- IOWA\_APN\_CONNECTION\_PROFILE\_RSC\_APN\_RATE\_CONTROL

Evaluation

### 3.7.3.6 iowa\_client\_get\_apn\_connection\_profile\_object\_link

#### Prototype

```
iowa_status_t iowa_client_get_apn_connection_profile_object_link(
    iowa_context_t contextP,
    const char *profileName,
    iowa_lwm2m_object_link_t *objectLinkP
);
```

**Description** `iowa_client_get_apn_connection_profile_object_link()` retrieves the LwM2M Object Link to an APN connection profile.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **profileName**

Unique name of the APN connection profile.

##### **objectLinkP**

Pointer to an `iowa_lwm2m_object_link_t` where to store the LwM2M Object Link to the APN connection profile.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

APN connection profile does not exist. Add first the profile with `iowa_client_add_apn_connection_profile()`.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

`profileName` or `objectLinkP` is nil.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

APN connection profile management was not enabled. Call first `iowa_client_enable_apn_connection_profile_object()`.

**Header File** `objects/iowa_apn_connection_profile.h`

**Notes** This function is useful to fill the `activatedProfileNamesList` field of the `iowa_cellular_connectivity_info_t` structure.

## 3.8 AT Command Object API

This Lwm2m object can be used to execute an AT command on a cellular modem.

To be able to use this object, `iowa_at_command.h` must be included.

### 3.8.1 Callbacks

#### 3.8.1.1 iowa\_at\_command\_run\_t

This callback is used to execute an AT command.

```
typedef iowa_status_t (*iowa_at_command_run_t)(iowa_sensor_t id,
                                              char *command,
                                              int timeout,
                                              void *userDataCallback,
                                              iowa_context_t contextP);
```

##### **id**

ID of the object.

##### **command**

The AT command to run.

##### **timeout**

Amount of time in seconds allowed for the modem to respond to the command.

##### **userDataCallback**

Application specific data from `iowa_client_add_at_command_object`. Can be nil.

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

**Return Value** IOWA\_COAP\_NO\_ERROR in case of success or an error status.

**Header File** `objects/iowa_at_command.h`



## 3.8.2 API

### 3.8.2.1 iowa\_client\_add\_at\_command\_object

#### Prototype

```
iowa_status_t iowa_client_add_at_command_object(iowa_context_t contextP,
                                                uint16_t optFlags,
                                                iowa_at_command_run_t run,
                                                void *userDataCallback,
                                                iowa_sensor_t *idP);
```

**Description** `iowa_client_add_at_command_object()` creates an AT Command object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to add optional resources.

##### **run**

Called to send an AT command to the modem.

##### **userDataCallback**

Application specific data pass to the callback. Can be nil.

##### **idP**

Used to store the ID of the object.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

no run callback provided means `run` is nil.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_at_command.h`

**Notes** Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- `IOWA_AT_COMMAND_RSC_TIMEOUT`

### 3.8.2.2 iowa\_client\_remove\_at\_command\_object

#### Prototype

```
iowa_status_t iowa_client_remove_at_command_object(iowa_context_t contextP,  
                                                    iowa_sensor_t id);
```

**Description** `iowa_client_remove_at_command_object()` removes an AT Command object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not an AT Command object. Valid `id` are only returned by `iowa_client_add_at_command_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

AT Command referred by `id` does not exist.

**Header File** `objects/iowa_at_command.h`

### 3.8.2.3 iowa\_client\_at\_command\_set\_response

#### Prototype

```
iowa_status_t iowa_client_at_command_set_response(iowa_context_t contextP,  
                                                  iowa_sensor_t id,  
                                                  const char *command,  
                                                  const char *response,  
                                                  const char *status);
```

**Description** `iowa_client_at_command_set_response()` updates result values after having executed an AT command.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object.

##### **command**

The executed AT command.

##### **response**

Response to the command.

##### **status**

Status of the command execution as returned by the modem.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not an AT Command object. Valid `id` are only returned by `iowa_client_add_at_command_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

AT Command referred by `id` does not exist.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_at_command.h`

**Notes** `response` and `status` can spread on multiple lines.

## 3.9 Bearer Selection Object API

This LwM2M object allows via remote bearer and network configuration to overwrite automatic network and bearer selection e.g. as supported by the UICC.

To be able to use this object, `iowa_bearer_selection.h` must be included.

### 3.9.1 Data Structures and Constants

#### 3.9.1.1 iowa\_bearer\_selection\_info\_t

```
typedef struct
{
    int            *preferredCommBearerList;
    uint16_t       preferredCommBearerNumber;
    int            acceptableGsm;
    int            acceptableUmts;
    int            acceptableLte;
    int            acceptableEvDo;
    char           *cellLockList;
    char           *operatorList;
    bool           operatorListMode;
    char           *availablePlmns;
    int            acceptableRsrpNbIot;
    int            plmnSearchTimer;
    bool           attachWoPdnConnection;
} iowa_bearer_selection_info_t;
```

##### **preferredCommBearer**

Preferred communications bearer.

##### **acceptableGsm**

Provides guide to the application when performing manual network selection.

##### **acceptableUmts**

Provides guide to the application when performing manual network selection.

##### **acceptableLte**

Provides guide to the application when performing manual network selection.

##### **acceptableEvDo**

Provides guide to the application when performing manual network selection.

##### **cellLockList**

List of allowed Global Cell Identities.

##### **operatorList**

List of MCC+MNC of operators, in priority order.

##### **operatorListMode**

Indicates whether resource operator list represents the allowed operator list (white list), or, the preferred operator list.

##### **availablePlmns**

Allows server to see results of network scan.

##### **acceptableRsrpNbIot**

Provides guide to the application when performing manual network selection.

##### **plmnSearchTimer**

Interval between periodic searches for higher priority PLMNs.

##### **attachWoPdnConnection**

0=attach with PDN connection, 1=attach without PDN connection

## 3.9.2 Callbacks

### 3.9.2.1 iowa\_bearer\_selection\_update\_state\_callback\_t

This callback is called when the Server writes new information on the Bearer selection object.

```
typedef iowa_status_t (*iowa_bearer_selection_update_state_callback_t) (
    iowa_sensor_t id,
    iowa_bearer_selection_info_t *infoP,
    void *userDataCallback,
    iowa_context_t contextP
);
```

***id***

The instance of the Bearer selection.

***infoP***

The bearer selection info.

***userDataCallback***

The user data callback.

***contextP***

The IOWA context.

**Return Value** IOWA\_COAP\_NO\_ERROR in case of success or an error status.

### 3.9.3 API

#### 3.9.3.1 iowa\_client\_add\_bearer\_selection\_object

##### Prototype

```
iowa_status_t iowa_client_add_bearer_selection_object(
    iowa_context_t contextP,
    uint16_t optFlags,
    iowa_bearer_selection_update_state_callback_t updateStateCallback,
    void *userDataCallback,
    iowa_sensor_t *idP
);
```

**Description** `iowa_client_add_bearer_selection_object()` creates a Bearer selection object.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **optFlags**

Optional flags to add optional resources.

###### **updateStateCallback**

Called to update state of the bearer selection. This is called when the server request a new state.

###### **userDataCallback**

Application specific data pass to the callback. Can be nil.

###### **idP**

Used to store the ID of the object.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- `optFlags` is equals to zero.
- no update state callback provided means `updateStateCallback` is nil.

###### **IOWA\_COAP\_409\_CONFLICT**

a bearer selection object already exists. To reconfigure the bearer selection object, call first `iowa_client_remove_bearer_selection_object()`.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_bearer_selection.h`

**Notes** Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

Since this object has no mandatory resource, at least one optional resource must be used. To add optional resources, you can use the following flags:

- `IOWA_BEARER_SELECTION_RSC_PREFERRED_COMM_BEARER`
- `IOWA_BEARER_SELECTION_RSC_ACCEPTABLE_RSSI_GSM`

- IOWA\_BEARER\_SELECTION\_RSC\_ACCEPTABLE\_RSCP\_UMTS
- IOWA\_BEARER\_SELECTION\_RSC\_ACCEPTABLE\_RSRP\_LTE
- IOWA\_BEARER\_SELECTION\_RSC\_ACCEPTABLE\_RSSI\_EV\_DO
- IOWA\_BEARER\_SELECTION\_RSC\_CELL\_LOCK\_LIST
- IOWA\_BEARER\_SELECTION\_RSC\_OPERATOR\_LIST
- IOWA\_BEARER\_SELECTION\_RSC\_OPERATOR\_LIST\_MODE
- IOWA\_BEARER\_SELECTION\_RSC\_AVAILABLE\_PLMNS
- IOWA\_BEARER\_SELECTION\_RSC\_ACCEPTABLE\_RSRP\_NB\_IOT
- IOWA\_BEARER\_SELECTION\_RSC\_PLMN\_SEARCH\_TIMER
- IOWA\_BEARER\_SELECTION\_RSC\_ATTACH\_WO\_PDN\_CONNECTION

Evaluation

### 3.9.3.2 iowa\_client\_remove\_bearer\_selection\_object

#### Prototype

```
iowa_status_t iowa_client_remove_bearer_selection_object(iowa_context_t contextP,  
                                                         iowa_sensor_t id);
```

**Description** `iowa_client_remove_bearer_selection_object()` removes a Bearer selection object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a bearer selection object. Valid `id` are only returned by `iowa_client_add_bearer_selection_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

bearer selection referred by `id` does not exist.

**Header File** `objects/iowa_bearer_selection.h`



### 3.9.3.3 iowa\_client\_bearer\_selection\_update

#### Prototype

```
iowa_status_t iowa_client_bearer_selection_update(iowa_context_t contextP,
                                                  iowa_sensor_t id,
                                                  uint16_t flags,
                                                  iowa_bearer_selection_info_t *infoP);
```

**Description** `iowa_client_bearer_selection_update()` updates the Bearer selection information.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object.

##### **flags**

Optional flags to update resources.

##### **info**

The Bearer selection information to update.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a bearer selection object. Valid `id` are only returned by `iowa_client_add_bearer_selection_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

bearer selection referred by `id` does not exist.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_bearer_selection.h`

**Notes** To specify resources to update, you can use the following flags:

- IOWA\_BEARER\_SELECTION\_RSC\_PREFERRED\_COMM\_BEARER
- IOWA\_BEARER\_SELECTION\_RSC\_ACCEPTABLE\_RSSI\_GSM
- IOWA\_BEARER\_SELECTION\_RSC\_ACCEPTABLE\_RSCP\_UMTS
- IOWA\_BEARER\_SELECTION\_RSC\_ACCEPTABLE\_RSRP\_LTE
- IOWA\_BEARER\_SELECTION\_RSC\_ACCEPTABLE\_RSSI\_EV\_DO
- IOWA\_BEARER\_SELECTION\_RSC\_CELL\_LOCK\_LIST
- IOWA\_BEARER\_SELECTION\_RSC\_OPERATOR\_LIST
- IOWA\_BEARER\_SELECTION\_RSC\_OPERATOR\_LIST\_MODE
- IOWA\_BEARER\_SELECTION\_RSC\_AVAILABLE\_PLMNS
- IOWA\_BEARER\_SELECTION\_RSC\_ACCEPTABLE\_RSRP\_NB\_IOT
- IOWA\_BEARER\_SELECTION\_RSC\_PLMN\_SEARCH\_TIMER
- IOWA\_BEARER\_SELECTION\_RSC\_ATTACH\_WO\_PDN\_CONNECTION

## 3.10 Cellular Connectivity Object API

This Lwm2M object specifies resources to enable a device to connect to a 3GPP or 3GPP2 bearer, including GPRS/EDGE, UMTS, LTE, NB-IoT, SMS.

To be able to use this object, `iowa_cellular_connectivity.h` must be included.

### 3.10.1 Data Structures and Constants

#### 3.10.1.1 `iowa_cellular_connectivity_info_t`

```
typedef struct
{
    iowa_lwm2m_object_link_t *activatedProfileNamesList;
    uint16_t                  activatedProfileNamesNumber;
    char                      *smc;
    int                       disableRadioPeriod;
    char                      *moduleActivationCode;
    int                       psmTimer;
    int                       activeTimer;
    int                       servingPlmnRateControl;
    char                      *edrxParamIuMode;
    char                      *edrxParamWbS1Mode;
    char                      *edrxParamNbS1Mode;
    char                      *edrxParamAGbmMode;
} iowa_cellular_connectivity_info_t;
```

##### ***activatedProfileNamesList***

list of links to instances of the APN connection profile object representing every APN connection profile that has an activated connection to a PDN.

##### ***activatedProfileNamesNumber***

number of links to instances of the APN connection profile object representing every APN connection profile that has an activated connection to a PDN.

##### ***smc***

address of the sms center.

##### ***disableRadioPeriod***

time period for which the device shall disconnect from cellular radio.

##### ***moduleActivationCode***

configurable in case the application needs to issue a code.

##### ***psmTimer***

Power Saving Mode timer.

##### ***activeTimer***

active timer.

##### ***servingPlmnRateControl***

maximum number of allowed uplink PDU transmissions.

##### ***edrxParamIuMode***

Extended DRX parameters for Iu mode.

##### ***edrxParamWbS1Mode***

Extended DRX parameters for WB-S1 mode.

##### ***edrxParamNbS1Mode***

Extended DRX parameters for NB-S1 mode.

##### ***edrxParamAGbmMode***

Extended DRX parameters for A/Gb mode.

## 3.10.2 Callbacks

### 3.10.2.1 iowa\_cellular\_connectivity\_update\_state\_callback\_t

This callback is called when the Server writes new information on the Cellular connectivity object.

```
typedef iowa_status_t (*iowa_cellular_connectivity_update_state_callback_t)(
    iowa_sensor_t id,
    iowa_cellular_connectivity_info_t *infoP,
    void *userDataCallback,
    iowa_context_t contextP
);
```

***id***

The instance of the Cellular connectivity.

***infoP***

The Cellular connectivity info.

***userDataCallback***

The user data callback.

***contextP***

The IOWA context.

**Return Value** IOWA\_COAP\_NO\_ERROR in case of success or an error status.

### 3.10.3 API

#### 3.10.3.1 iowa\_client\_add\_cellular\_connectivity\_object

##### Prototype

```
iowa_status_t iowa_client_add_cellular_connectivity_object(
    iowa_context_t contextP,
    uint16_t optFlags,
    iowa_cellular_connectivity_update_state_callback_t updateStateCallback,
    void *userDataCallback,
    iowa_sensor_t *idP
);
```

**Description** `iowa_client_add_cellular_connectivity_object()` creates a Cellular connectivity object.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **optFlags**

Optional flags to add optional resources.

###### **updateStateCallback**

Called to update state of the cellular connectivity. This is called when the server request a new state.

###### **userDataCallback**

Application specific data pass to the callback. Can be nil.

###### **idP**

Used to store the ID of the object.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

no update state callback provided means `updateStateCallback` is nil.

###### **IOWA\_COAP\_409\_CONFLICT**

a cellular connectivity object already exists. Call first `iowa_client_remove_cellular_connectivity_object()`.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_cellular_connectivity.h`

**Notes** Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_SMSC\_ADDRESS
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_DISABLE\_RADIO\_PERIOD
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_MODULE\_ACTIVATION\_CODE
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_PSM\_TIMER
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_ACTIVE\_TIMER
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_PLMN\_RATE\_CONTROL
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_EDRX\_PARAM\_IU\_MODE

- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_EDRX\_PARAM\_WB\_S1\_MODE
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_EDRX\_PARAM\_NB\_S1\_MODE
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_EDRX\_PARAM\_A\_GB\_MODE

Evaluation

### 3.10.3.2 iowa\_client\_remove\_cellular\_connectivity\_object

#### Prototype

```
iowa_status_t iowa_client_remove_cellular_connectivity_object(iowa_context_t contextP,  
                                                             iowa_sensor_t id);
```

**Description** `iowa_client_remove_cellular_connectivity_object()` removes a Cellular connectivity object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a cellular connectivity object. Valid `id` are only returned by `iowa_client_add_cellular_connectivity_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

cellular connectivity referred by **id** does not exist.

**Header File** `objects/iowa_cellular_connectivity.h`

### 3.10.3.3 iowa\_client\_cellular\_connectivity\_update

#### Prototype

```
iowa_status_t iowa_client_cellular_connectivity_update(
    iowa_context_t contextP,
    iowa_sensor_t id,
    uint16_t flags,
    iowa_cellular_connectivity_info_t *infoP
);
```

**Description** `iowa_client_cellular_connectivity_update()` updates the Cellular connectivity information.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object.

##### **flags**

Optional flags to update resources.

##### **infoP**

The Cellular connectivity information to update.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a cellular connectivity object. Valid `id` are only returned by `iowa_client_add_cellular_connectivity_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

cellular connectivity referred by `id` does not exist.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_cellular_connectivity.h`

**Notes** To specify resources to update, you can use the following flags:

- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_SMSC\_ADDRESS
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_DISABLE\_RADIO\_PERIOD
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_MODULE\_ACTIVATION\_CODE
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_PSM\_TIMER
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_ACTIVE\_TIMER
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_PLMN\_RATE\_CONTROL
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_EDRX\_PARAM\_IU\_MODE
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_EDRX\_PARAM\_WB\_S1\_MODE
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_EDRX\_PARAM\_NB\_S1\_MODE
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_EDRX\_PARAM\_A\_GB\_MODE
- IOWA\_CELLULAR\_CONNECTIVITY\_RSC\_ACTIVATED\_PROFILE\_NAMES

## 3.11 Connectivity Monitoring Object API

This LwM2M Object enables monitoring of parameters related to network connectivity.

To be able to use this object, `iowa_connectivity_monitoring.h` must be included.

### 3.11.1 Data Structures and Constants

#### 3.11.1.1 iowa\_connectivity\_monitoring\_info\_t

```
typedef struct
{
    int         networkBearer;
    int         *availableNetworkBearerList;
    uint16_t    availableNetworkBearerNumber;
    int         radioSignalStrength;
    int         linkQuality;
    char        **ipAddressList;
    uint16_t    ipAddressNumber;
    char        **routerIpAddressesList;
    uint16_t    routerIpAddressesNumber;
    int         linkUtilization;
    char        **apnList;
    uint16_t    apnNumber;
    int         cellId;
    int         smnc;
    int         smcc;
} iowa_connectivity_monitoring_info_t;
```

##### **networkBearer**

Network bearer used for the current session.

##### **availableNetworkBearerList**

List of current available network bearers.

##### **availableNetworkBearerNumber**

Number of current available network bearers.

##### **radioSignalStrength**

Average value of the received signal strength indication.

##### **linkQuality**

Received link quality.

##### **ipAddressList**

List of IP addresses assigned to the connectivity interface.

##### **ipAddressNumber**

Number of IP addresses assigned to the connectivity interface.

##### **routerIpAddressesList**

List of IP addresses of the next-hop IP router.

##### **routerIpAddressesNumber**

Number of IP addresses of the next-hop IP router.

##### **linkUtilization**

The percentage indicating the average utilization of the link to the next-hop IP router.

##### **apnList**

List of Access Point Names.

##### **apnNumber**

Number of Access Point Names.



**cellId**

Serving Cell ID.

**smnc**

Serving Mobile Network Code.

**smcc**

Serving Mobile Country Code.

Evaluation

## 3.11.2 API

### 3.11.2.1 iowa\_client\_add\_connectivity\_monitoring\_object

#### Prototype

```
iowa_status_t iowa_client_add_connectivity_monitoring_object(iowa_context_t contextP,
                                                            uint16_t optFlags,
                                                            iowa_sensor_t *idP);
```

**Description** `iowa_client_add_connectivity_monitoring_object()` creates a Connectivity monitoring object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to add optional resources.

##### **idP**

Used to store the ID of the object.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_409\_CONFLICT**

a connectivity monitoring object already exists. Call first `iowa_client_remove_connectivity_monitoring_object()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_connectivity_monitoring.h`

**Notes** Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_LINK\_QUALITY
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_ROUTER\_IP\_ADDR
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_LINK\_USAGE
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_APN
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_CELL\_ID
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_SMNC
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_SMCC

### 3.11.2.2 iowa\_client\_remove\_connectivity\_monitoring\_object

#### Prototype

```
iowa_status_t iowa_client_remove_connectivity_monitoring_object(iowa_context_t contextP,  
                                                                iowa_sensor_t id);
```

**Description** `iowa_client_remove_connectivity_monitoring_object()` removes a Connectivity monitoring object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a connectivity monitoring object. Valid `id` are only returned by `iowa_client_add_connectivity_monitoring_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

connectivity monitoring referred by `id` does not exist.

**Header File** `objects/iowa_connectivity_monitoring.h`

### 3.11.2.3 iowa\_client\_connectivity\_monitoring\_update

#### Prototype

```
iowa_status_t iowa_client_connectivity_monitoring_update(
    iowa_context_t contextP,
    iowa_sensor_t id,
    uint16_t flags,
    iowa_connectivity_monitoring_info_t *infoP
);
```

**Description** `iowa_client_connectivity_monitoring_update()` updates the Connectivity monitoring information.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object.

##### **flags**

Optional flags to update resources.

##### **infoP**

The Connectivity monitoring information to update.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a connectivity monitoring object. Valid `id` are only returned by `iowa_client_add_connectivity_monitoring_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

connectivity monitoring referred by `id` does not exist.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_connectivity_monitoring.h`

**Notes** To specify resources to update, you can use the following flags:

- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_BEARER
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_AVAILABLE\_BEARER
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_SIGNAL\_STRENGTH
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_LINK\_QUALITY
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_IP\_ADDR
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_ROUTER\_IP\_ADDR
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_LINK\_USAGE
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_APN
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_CELL\_ID
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_SMNC
- IOWA\_CONNECTIVITY\_MONITORING\_RSC\_SMCC

## 3.12 Connectivity Statistics Object API

This LwM2M Object enables client to collect statistical information and enables the LwM2M Server to retrieve these information, set the collection duration and reset the statistical parameters.

To be able to use this object, `iowa_connectivity_stats.h` must be included.

### 3.12.1 iowa\_client\_add\_connectivity\_stats\_object

#### Prototype

```
iowa_status_t iowa_client_add_connectivity_stats_object(iowa_context_t context,
                                                       uint16_t optFlags,
                                                       iowa_sensor_t *idP);
```

#### Description

`iowa_client_add_connectivity_stats_object()` creates a connectivity statistics object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to add optional resources.

##### **idP**

Used to store the ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_409\_CONFLICT**

a connectivity statistics object already exists. To reconfigure the connectivity statistics object, call first `iowa_client_remove_connectivity_stats_object()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`objects/iowa_connectivity_stats.h`

#### Notes

Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- IOWA\_CONNECTIVITY\_STATS\_RSC\_SMS\_TX\_COUNTER
- IOWA\_CONNECTIVITY\_STATS\_RSC\_SMS\_RX\_COUNTER
- IOWA\_CONNECTIVITY\_STATS\_RSC\_TX\_DATA
- IOWA\_CONNECTIVITY\_STATS\_RSC\_RX\_DATA
- IOWA\_CONNECTIVITY\_STATS\_RSC\_MAX\_MESSAGE\_SIZE

- IOWA\_CONNECTIVITY\_STATS\_RSC\_AVERAGE\_MESSAGE\_SIZE
- IOWA\_CONNECTIVITY\_STATS\_RSC\_COLLECTION\_PERIOD

Moreover, you can add several optional resources at one time by using the following flags:

- IOWA\_CONNECTIVITY\_STATS\_SMS
- IOWA\_CONNECTIVITY\_STATS\_IP\_DATA

Evaluation

### 3.12.2 iowa\_client\_remove\_connectivity\_stats\_object

#### Prototype

```
iowa_status_t iowa_client_remove_connectivity_stats_object(iowa_context_t contextP,  
                                                         iowa_sensor_t id);
```

#### Description

`iowa_client_remove_connectivity_stats_object()` removes a connectivity statistics object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a connectivity statistics object. Valid `id` are only returned by `iowa_client_add_connectivity_stats_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

no connectivity statistics object to remove. `iowa_client_add_connectivity_stats_object()` was not called before, or failed.

#### Header File

objects/iowa\_connectivity\_stats.h

### 3.12.3 iowa\_client\_connectivity\_stats\_update\_sms

#### Prototype

```
iowa_status_t iowa_client_connectivity_stats_update_sms(iowa_context_t context,
                                                         iowa_sensor_t id,
                                                         uint8_t direction);
```

#### Description

`iowa_client_connectivity_stats_update_sms()` updates the SMS TX or RX statistics.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

##### **direction**

Specify if this is a reception or a transmission trigger.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

Bad value for argument *direction* or *id* is not a connectivity statistics object. Valid *id* are only returned by `iowa_client_add_connectivity_stats_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

no connectivity statistics object added.

#### Header File

`objects/iowa_connectivity_stats.h`

#### Notes

Argument *direction* of `iowa_client_connectivity_stats_update_sms()` can be one of the following values:

- IOWA\_CONNECTIVITY\_STATS\_TX (0)
- IOWA\_CONNECTIVITY\_STATS\_RX (1)



### 3.12.4 iowa\_client\_connectivity\_stats\_update\_ip\_data

#### Prototype

```
iowa_status_t iowa_client_connectivity_stats_update_ip_data(iowa_context_t context,
                                                            iowa_sensor_t id,
                                                            uint8_t direction,
                                                            size_t length);
```

#### Description

`iowa_client_connectivity_stats_update_ip_data()` updates the IP data statistics.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object.

##### **direction**

Specify if this is a reception or a transmission trigger.

##### **length**

Length in bytes of the transmitted or received data.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

Bad value for argument *direction* or *id* is not a connectivity statistics object. Valid *id* are only returned by `iowa_client_add_connectivity_stats_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

no connectivity statistics object added.

#### Header File

`objects/iowa_connectivity_stats.h`

#### Notes

Argument *direction* of `iowa_client_connectivity_stats_update_ip_data()` can be one of the following values:

- `IOWA_CONNECTIVITY_STATS_TX` (0)
- `IOWA_CONNECTIVITY_STATS_RX` (1)

### 3.13 Digital Output Object API

This IPSO object represents generic digital output for non-specific actuators.

To be able to use this object, `iowa_digital_output.h` must be included.

#### 3.13.1 Callbacks

##### 3.13.1.1 `iowa_digital_output_state_callback_t`

This callback is used to update the state of the digital output. Request from a server to a client.

```
typedef iowa_status_t (*iowa_digital_output_state_callback_t)(iowa_sensor_t id,
                                                             bool state,
                                                             bool polarity,
                                                             void *userDataCallback,
                                                             iowa_context_t contextP);
```

***id***

ID of the object

***state***

New state

***polarity***

New polarity

***userDataCallback***

Application specific data from `iowa_client_add_digital_output_object`. Can be nil.

***contextP***

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

**Return Value** IOWA\_COAP\_NO\_ERROR in case of success or an error status.

### 3.13.2 API

#### 3.13.2.1 iowa\_client\_add\_digital\_output\_object

##### Prototype

```
iowa_status_t iowa_client_add_digital_output_object(
    iowa_context_t context,
    uint16_t optFlags,
    iowa_digital_output_state_callback_t updateStateCallback,
    void *userDataCallback,
    const char *applicationType,
    iowa_sensor_t *idP
);
```

**Description** `iowa_client_add_digital_output_object()` creates a digital output object.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **optFlags**

Optional flags to add optional resources.

###### **updateStateCallback**

Called to update state of the digital output. This is called when the server request a new state.

###### **userDataCallback**

Application specific data pass to the callback. Can be nil.

###### **applicationType**

The application type

###### **idP**

Used to store the ID of the object

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

no update state callback provided means *updateStateCallback* is nil.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_digital_output.h`

**Notes** Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- IOWA\_DIGITAL\_OUTPUT\_STATS\_RSC\_POLARITY

### 3.13.2.2 iowa\_client\_remove\_digital\_output\_object

#### Prototype

```
iowa_status_t iowa_client_remove_digital_output_object(iowa_context_t context,
                                                       iowa_sensor_t id);
```

**Description** `iowa_client_remove_digital_output_object()` removes a digital output object.

#### Arguments

##### **context**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a digital output object. Valid `id` are only returned by `iowa_client_add_digital_output_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

digital output referred by `id` does not exist.

**Header File** `objects/iowa_digital_output.h`

### 3.13.2.3 iowa\_client\_digital\_output\_update\_state

#### Prototype

```
iowa_status_t iowa_client_digital_output_update_state(iowa_context_t context,
                                                    iowa_sensor_t id,
                                                    bool state,
                                                    bool polarity);
```

**Description** `iowa_client_digital_output_update_state()` updates values of a digital output object.

#### Arguments

##### **context**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

##### **state**

New state

##### **polarity**

New polarity

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a digital output object. Valid `id` are only returned by `iowa_client_add_digital_output_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

digital output referred by `id` does not exist.

**Header File** `objects/iowa_digital_output.h`

## 3.14 Firmware Update Object API

This Lwm2m Object enables management of firmware which is to be updated.

To be able to use this object, `iowa_firmware_update.h` must be included and **IOWA\_SUPPORT\_FIRMWARE\_UPDATE\_OBJECT** must be defined before building the library.

The [Device Update][Device Update]/[Firmware Update][Firmware Update] part of this specification adds more explanation about its mechanism and how to use it with IOWA.

### 3.14.1 Data Structures and Constants

#### 3.14.1.1 `iowa_fw_status_t`

This is an enumeration of the following values:

**IOWA\_FW\_STATUS\_SUCCESSFUL**

success of the new firmware package download or of the firmware update.

**IOWA\_FW\_STATUS\_OUT\_OF\_STORAGE**

not enough storage for the new firmware package. (*downloadCb* only)

**IOWA\_FW\_STATUS\_OUT\_OF\_MEMORY**

out of memory error during the download of the new firmware package. (*downloadCb* only)

**IOWA\_FW\_STATUS\_CONNECTION\_LOST**

connection lost during the download of the new firmware package. (*downloadCb* only)

**IOWA\_FW\_STATUS\_INTEGRITY\_CHECK\_FAILURE**

integrity check failure of the new firmware package.

**IOWA\_FW\_STATUS\_UNSUPPORTED\_TYPE**

unsupported new firmware package type.

**IOWA\_FW\_STATUS\_INVALID\_URI**

invalid URI to download the new firmware package. (*downloadCb* only)

**IOWA\_FW\_STATUS\_UPDATE\_FAILED**

firmware update failed. (*updateCb* only)

**IOWA\_FW\_STATUS\_UNSUPPORTED\_PROTOCOL**

unsupported protocol in URI to download the new firmware package. (*downloadCb* only)

## 3.14.2 Callbacks

### 3.14.2.1 iowa\_fw\_download\_callback\_t

This callback is called when the Server requests the device to download a new Firmware Package.

```
typedef void(*iowa_fw_download_callback_t) (char * uri,  
                                           void * userData,  
                                           iowa_context_t contextP);
```

#### **uri**

The URI to download the package from.

#### **userData**

The parameter to `iowa_client_firmware_update_configure()`.

#### **contextP**

The IOWA context on which `iowa_client_firmware_update_configure()` was called.

*uri* can be nil. In this case, a current download must be aborted.

### 3.14.2.2 iowa\_fw\_write\_callback\_t

This callback is called several times when the Server pushes the new Firmware Package to the device. The expected behavior is the same as writing to a file stream i.e. unless it is reset, written data are appended to the previous ones.

```
typedef iowa_fw_status_t(*iowa_fw_write_callback_t) (iowa_fw_write_cmd_t cmd,
                                                    size_t dataLength,
                                                    uint8_t *data,
                                                    void *userData,
                                                    iowa_context_t contextP);
```

At the start of the push of the Firmware Package or if the LwM2M Server cancels it, this callback is called with the following parameters:

**cmd**

**IOWA\_FW\_PACKAGE\_RESET**

**dataLength**

0

**data**

NULL

**userData**

The parameter to `iowa_client_firmware_update_configure()`.

**contextP**

The IOWA context on which `iowa_client_firmware_update_configure()` was called.

When the Firmware Package is received, this callback is called several times with the following parameters:

**cmd**

**IOWA\_FW\_PACKAGE\_WRITE**

**dataLength**

The length of the buffer pointed by *data*.

**data**

The next chunk of the Firmware Package to write.

**userData**

The parameter to `iowa_client_firmware_update_configure()`.

**contextP**

The IOWA context on which `iowa_client_firmware_update_configure()` was called.

At the end of the push of the Firmware package, this callback is called with the following parameters:

**cmd**

**IOWA\_FW\_PACKAGE\_WRITE**

**dataLength**

0

**data**

NULL

**userDataP**

The data passed to `iowa_client_firmware_update_configure()`.

**contextP**

The IOWA context on which `iowa_client_firmware_update_configure()` was called.

**Return Value**

**IOWA\_FW\_STATUS\_SUCCESSFUL**

success.



**IOWA\_FW\_STATUS\_OUT\_OF\_STORAGE**

not enough storage for the new Firmware Package.

**IOWA\_FW\_STATUS\_OUT\_OF\_MEMORY**

out of memory error.

**IOWA\_FW\_STATUS\_INTEGRITY\_CHECK\_FAILURE**

integrity check failure of the new Firmware Package.

**IOWA\_FW\_STATUS\_UNSUPPORTED\_TYPE**

unsupported new Firmware Package type.

Evaluation

### 3.14.2.3 iowa\_fw\_update\_callback\_t

This callback is called when the Server requests the device to update itself with the new Firmware Package.

```
typedef void(*iowa_fw_update_callback_t) (void * userData,  
                                          iowa_context_t contextP);
```

#### ***userData***

The parameter to `iowa_client_firmware_update_configure()`.

#### ***contextP***

The IOWA context on which `iowa_client_firmware_update_configure()` was called.

Evaluation

### 3.14.3 API

#### 3.14.3.1 iowa\_client\_firmware\_update\_configure

##### Prototype

```
iowa_status_t iowa_client_firmware_update_configure(iowa_context_t contextP,  
                                                    const char *packageName,  
                                                    const char *packageVersion,  
                                                    iowa_fw_download_callback_t downloadCb  
                                                    ,  
                                                    iowa_fw_write_callback_t writeCb,  
                                                    iowa_fw_update_callback_t updateCb,  
                                                    void *userData);
```

**Description** `iowa_client_firmware_update_configure()` configures the firmware update feature of the IOWA stack.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **packageName**

The user-defined name of the current firmware. This can be nil.

###### **packageVersion**

The user-defined version of the current firmware. This can be nil.

###### **downloadCb**

The callback called to download a new firmware. This can be nil.

###### **writeCb**

The callback called to write chunks of the new firmware to the device storage. This can be nil.

###### **updateCb**

The callback called to update the device with the new firmware.

###### **userData**

Passed as argument to the callbacks.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

###### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- `updateCb` is nil.
- both `downloadCb` and `writeCb` are nil. At least one must be defined.

###### **IOWA\_COAP\_409\_CONFLICT**

the firmware update feature is already configured.

**Header File** `objects/iowa_firmware_update.h`

**Notes** The LwM2M Server has two methods to provide the Firmware Package:

- the pull method: the LwM2M Server provides the URI of the Firmware Package and the LwM2M Client downloads it directly. To use this method, *downloadCb* must be set.
- the push method: the LwM2M Server writes the Firmware Package in a LwM2M Resource exposed by the Client. To use this method, *writeCb* must be set.

The Client can support both methods at the same time.

*downloadCb* and *updateCb* do not return any value. The progress and result of their operation are indicated asynchronously by calling `iowa_client_firmware_update_set_status()`.

Evaluation

### 3.14.3.2 iowa\_client\_firmware\_update\_configure\_full

#### Prototype

```
iowa_status_t iowa_client_firmware_update_configure_full(
    iowa_context_t contextP,
    const char *packageName,
    const char *packageVersion,
    uint8_t protocolSupport,
    iowa_fw_download_callback_t downloadCb,
    iowa_fw_write_callback_t writeCb,
    iowa_fw_update_callback_t updateCb,
    void *userData
);
```

**Description** `iowa_client_firmware_update_configure_full()` configures the firmware update feature of the IOWA stack with the “protocol support” resource.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **packageName**

The user-defined name of the current firmware. This can be nil.

##### **packageVersion**

The user-defined version of the current firmware. This can be nil.

##### **protocolSupport**

A bit-mask indicating supported protocols in `downloadCb`. This can be 0.

##### **downloadCb**

The callback called to download a new firmware. This can be nil.

##### **writeCb**

The callback called to write chunks of the new firmware to the device storage. This can be nil.

##### **updateCb**

The callback called to update the device with the new firmware.

##### **userData**

Passed as argument to the callbacks.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- `updateCb` is nil.
- both `downloadCb` and `writeCb` are nil.
- `downloadCb` is nil and `protocolSupport` is not 0.

##### **IOWA\_COAP\_409\_CONFLICT**

the firmware update feature is already configured.

**Header File** `objects/iowa_firmware_update.h`

**Notes** *protocolSupport* is a combination of the following:

- **IOWA\_FIRMWARE\_UPDATE\_PROTOCOL\_SUPPORT\_COAP**: Constrained Application Protocol (CoAP)
- **IOWA\_FIRMWARE\_UPDATE\_PROTOCOL\_SUPPORT\_COAPS**: DTLS-Secured CoAP
- **IOWA\_FIRMWARE\_UPDATE\_PROTOCOL\_SUPPORT\_HTTP**: HTTP 1.1
- **IOWA\_FIRMWARE\_UPDATE\_PROTOCOL\_SUPPORT\_HTTPS**: TLS-Secured HTTP 1.1
- **IOWA\_FIRMWARE\_UPDATE\_PROTOCOL\_SUPPORT\_COAP\_TCP**: CoAP over TCP
- **IOWA\_FIRMWARE\_UPDATE\_PROTOCOL\_SUPPORT\_COAP\_TLS**: CoAP over TLS

Evaluation

### 3.14.3.3 iowa\_client\_firmware\_update\_set\_status

#### Prototype

```
iowa_status_t iowa_client_firmware_update_set_status(iowa_context_t contextP,  
                                                    iowa_fw_status_t status);
```

**Description** `iowa_client_firmware_update_set_status()` informs the IOWA stack of the result of the callbacks `downloadCb` and `updateCb` of `iowa_client_firmware_update_configure()`.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **status**

The result of the current firmware update operation.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

the firmware update feature is not configured. Call first `iowa_client_firmware_update_configure()`.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

the value of `status` is unexpected. See `iowa_fw_status_t` for the possible value depending of the context.

**Header File** `objects/iowa_firmware_update.h`

## 3.15 GPS Object API

This IPSO object represents GPS coordinates.

To be able to use this object, `iowa_gps.h` must be included.

### 3.15.1 iowa\_client\_add\_gps\_object

#### Prototype

```
iowa_status_t iowa_client_add_gps_object(iowa_context_t context,
                                         uint16_t optFlags,
                                         const char *applicationType,
                                         iowa_sensor_t *idP);
```

#### Description

`iowa_client_add_gps_object()` creates a GPS object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to add optional resources.

##### **applicationType**

The application type

##### **idP**

Used to store the ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`objects/iowa_gps.h`

#### Notes

Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- IOWA\_GPS\_RSC\_UNCERTAINTY
- IOWA\_GPS\_RSC\_COMPASS\_DIRECTION
- IOWA\_GPS\_RSC\_VELOCITY
- IOWA\_GPS\_RSC\_TIMESTAMP



### 3.15.2 iowa\_client\_remove\_gps\_object

#### Prototype

```
iowa_status_t iowa_client_remove_gps_object(iowa_context_t context,
                                             iowa_sensor_t id);
```

#### Description

`iowa_client_remove_gps_object()` removes a GPS object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a GPS object. Valid `id` are only returned by `iowa_client_add_gps_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

GPS referred by `id` does not exist.

#### Header File

`objects/iowa_gps.h`

### 3.15.3 iowa\_client\_gps\_update\_location

#### Prototype

```
iowa_status_t iowa_client_gps_update_location(iowa_context_t context,
                                              iowa_sensor_t id,
                                              const char *latitude,
                                              const char *longitude);
```

#### Description

`iowa_client_gps_update_location()` updates values of a GPS object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

##### **latitude**

New latitude

##### **longitude**

New longitude

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a GPS object. Valid `id` are only returned by `iowa_client_add_gps_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

GPS referred by **id** does not exist.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- `iowa_system_gettime()` returned an error.

#### Header File

`objects/iowa_gps.h`

### 3.15.4 iowa\_client\_gps\_update\_location\_full

#### Prototype

```
iowa_status_t iowa_client_gps_update_location_full(iowa_context_t context,
                                                    iowa_sensor_t id,
                                                    const char *latitude,
                                                    const char *longitude,
                                                    const char *uncertainty,
                                                    float compassDirection,
                                                    size_t velocityLength,
                                                    uint8_t *velocity);
```

#### Description

`iowa_client_gps_update_location_full()` updates values of a GPS object. Optional resources are included.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

##### **latitude**

New latitude

##### **longitude**

New longitude

##### **uncertainty**

The accuracy of the position in meters.

##### **compassDirection**

Measured Direction between 0 and 360 deg.

##### **velocityLength**

Length of the velocity array

##### **velocity**

The velocity of the device as defined in 3GPP 23.032 GAD specification.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a GPS object. Valid `id` are only returned by `iowa_client_add_gps_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

GPS referred by `id` does not exist.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

`compassDirection`'s value is outside the [0.0, 360.0] range.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- `iowa_system_gettime()` returned an error.

## Header File

objects/iowa\_gps.h

Evaluation

## 3.16 Gyrometer Object API

This IPSO Object is used to report the current reading of a gyrometer sensor in 3 axes.

To be able to use this object, `iowa_gyrometer.h` must be included.

### 3.16.1 iowa\_client\_add\_gyrometer\_object

#### Prototype

```
iowa_status_t iowa_client_add_gyrometer_object(iowa_context_t context,
                                              uint16_t optFlags,
                                              float minRangeValue,
                                              float maxRangeValue,
                                              const char *sensorUnits,
                                              const char *applicationType,
                                              iowa_sensor_t *idP);
```

#### Description

`iowa_client_add_gyrometer_object()` creates a gyrometer object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to add optional resources.

##### **minRangeValue**

Minimal range value for the gyrometer.

##### **maxRangeValue**

Maximal range value for the gyrometer.

##### **sensorUnits**

Measurement units definition

##### **applicationType**

The application type

##### **idP**

Used to store the ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`minRangeValue` argument is superior to `maxRangeValue` argument.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`objects/iowa_gyrometer.h`

## Notes

Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- IOWA\_GYROMETER\_RSC\_Y\_VALUE
- IOWA\_GYROMETER\_RSC\_Z\_VALUE
- IOWA\_GYROMETER\_RSC\_MIN\_X\_VALUE
- IOWA\_GYROMETER\_RSC\_MAX\_X\_VALUE
- IOWA\_GYROMETER\_RSC\_MIN\_Y\_VALUE
- IOWA\_GYROMETER\_RSC\_MAX\_Y\_VALUE
- IOWA\_GYROMETER\_RSC\_MIN\_Z\_VALUE
- IOWA\_GYROMETER\_RSC\_MAX\_Z\_VALUE
- IOWA\_GYROMETER\_RSC\_RESET\_MIN\_MAX\_VALUES
- IOWA\_GYROMETER\_RSC\_MIN\_RANGE\_VALUE
- IOWA\_GYROMETER\_RSC\_MAX\_RANGE\_VALUE

Moreover, you can add several optional resources at one time by using the following flags:

- IOWA\_GYROMETER\_3\_AXIS
- IOWA\_GYROMETER\_MIN\_MAX\_VALUES
- IOWA\_GYROMETER\_RANGE\_VALUE

### 3.16.2 iowa\_client\_remove\_gyrometer\_object

#### Prototype

```
iowa_status_t iowa_client_remove_gyrometer_object(iowa_context_t context,  
                                                  iowa_sensor_t id);
```

#### Description

`iowa_client_remove_gyrometer_object()` removes a gyrometer object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a gyrometer object. Valid `id` are only returned by `iowa_client_add_gyrometer_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

gyrometer referred by `id` does not exist.

#### Header File

`objects/iowa_gyrometer.h`

### 3.16.3 iowa\_client\_gyrometer\_update\_axis

#### Prototype

```
iowa_status_t iowa_client_gyrometer_update_axis(iowa_context_t context,
                                                iowa_sensor_t id,
                                                float xValue,
                                                float yValue,
                                                float zValue);
```

#### Description

`iowa_client_gyrometer_update_axis()` updates values of a gyrometer object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

##### **xValue**

X value axis

##### **yValue**

Y value axis

##### **zValue**

Z value axis

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a gyrometer object. Valid `id` are only returned by `iowa_client_add_gyrometer_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

gyrometer referred by `id` does not exist.

#### Header File

`objects/iowa_gyrometer.h`



## 3.17 IPSO Objects

This part allows the possibility to manipulate several IPSO Objects.

To be able to use these objects, `iowa_ipso.h` must be included.

### 3.17.1 `iowa_client_IPSO_add_sensor`

#### Prototype

```
iowa_status_t iowa_client_IPSO_add_sensor(iowa_context_t contextP,
                                          iowa_IPSO_ID_t type,
                                          float value,
                                          const char * unit,
                                          const char * appType,
                                          float rangeMin, float rangeMax,
                                          iowa_sensor_t * idP);
```

#### Description

`iowa_client_IPSO_add_sensor()` adds a new IPSO sensor for the LwM2M Client to handle. The sensor is defined by its type.

The unit, the application type and the range are only informative and reported as-is to the LwM2M Server. Note that the LwM2M Server can modify the application type.

#### Arguments

##### ***contextP***

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### ***type***

The type of sensor. See below.

##### ***value***

The initial value measured by the sensor.

##### ***unit***

The unit of the measured value as a nil-terminated string. This can be nil.

##### ***appType***

The application type of the sensor as a free-form nil-terminated string. This can be nil.

##### ***rangeMin***

The minimum value that can be measured by the sensor.

##### ***rangeMax***

The maximum value that can be measured by the sensor.

##### ***idP***

Used to store the ID of the created sensor. Not checked at runtime.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

*type* is unknown.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- the sensor uses a Boolean value and *value* is neither 0.0 nor 1.0.
- the sensor uses a percentage value and *value* is outside the [0.0, 100.0] range.
- the sensor uses a compass direction value and *value* is outside the [0.0, 360.0] range.

### IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR

a memory allocation or a call to `iowa_system_gettime()` failed.

### Header File

objects/iowa\_ipso.h

### Notes

If both *rangeMin* and *rangeMax* are set to zero, the matching resources are ignored in the LwM2M Object.

*unit* is not duplicated nor freed by IOWA. Make sure it is available until `iowa_close()` or `iowa_client_IPSO_remove_sensor()` is called. It is advised to use static strings.

*appType* is duplicated internally by IOWA and can be reused or freed by the caller.

Only a call to `iowa_client_IPSO_remove_sensor()` can free the memory allocated by `iowa_client_IPSO_add_sensor()`.

### iowa\_IPSO\_ID\_t

This is an enumeration of the LwM2M IDs of the supported sensor types. See below.

#### Float value sensors

- IOWA\_IPSO\_ANALOG\_INPUT (3202)
- IOWA\_IPSO\_GENERIC (3300)
- IOWA\_IPSO\_ILLUMINANCE (3301)
- IOWA\_IPSO\_TEMPERATURE (3303)
- IOWA\_IPSO\_HUMIDITY (3304)
- IOWA\_IPSO\_BAROMETER (3315)
- IOWA\_IPSO\_VOLTAGE (3316)
- IOWA\_IPSO\_CURRENT (3317)
- IOWA\_IPSO\_FREQUENCY (3318)
- IOWA\_IPSO\_DEPTH (3319)
- IOWA\_IPSO\_PERCENTAGE (3320)
- IOWA\_IPSO\_ALTITUDE (3321)
- IOWA\_IPSO\_LOAD (3322)
- IOWA\_IPSO\_PRESSURE (3323)
- IOWA\_IPSO\_LOUDNESS (3324)
- IOWA\_IPSO\_CONCENTRATION (3325)
- IOWA\_IPSO\_ACIDITY (3326)
- IOWA\_IPSO\_CONDUCTIVITY (3327)
- IOWA\_IPSO\_POWER (3328)
- IOWA\_IPSO\_POWER\_FACTOR (3329)
- IOWA\_IPSO\_RATE (3346)
- IOWA\_IPSO\_DISTANCE (3330)
- IOWA\_IPSO\_ENERGY (3331)

**Boolean value sensors** For these sensors, the value must be either 0.0 or 1.0:

- IOWA\_IPSO\_DIGITAL\_INPUT (3200)
- IOWA\_IPSO\_PRESENCE (3302)
- IOWA\_IPSO\_ON\_OFF\_SWITCH (3342)
- IOWA\_IPSO\_PUSH\_BUTTON (3347)

**Percentage value sensors** For these sensors, the value must be between 0.0 and 100.0:

- IOWA\_IPSO\_LEVEL\_CONTROL (3343)

**Compass direction value sensors** For these sensors, the value must be between 0.0 and 360.0:

- IOWA\_IPSO\_DIRECTION (3332)

Evaluation

### 3.17.2 iowa\_client\_IPSO\_update\_value

#### Prototype

```
iowa_status_t iowa_client_IPSO_update_value(iowa_context_t contextP,  
                                             iowa_sensor_t id,  
                                             float value);
```

#### Description

`iowa_client_IPSO_update_value()` updates the value of an IPSO sensor.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

The ID of the sensor as returned by `iowa_client_IPSO_add_sensor()`.

##### **value**

The new value measured by the sensor.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`id` does not match any known sensor.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- the sensor uses a Boolean value and `value` is neither 0.0 nor 1.0.
- the sensor uses a percentage value and `value` is outside the [0.0, 100.0] range.
- the sensor uses a compass direction value and `value` is outside the [0.0, 360.0] range.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation or a call to `iowa_system_gettime()` failed.

#### Header File

objects/iowa\_ipso.h

### 3.17.3 iowa\_client\_IPSO\_update\_values

#### Prototype

```
iowa_status_t iowa_client_IPSO_update_values(iowa_context_t contextP,
                                             iowa_sensor_t id,
                                             size_t valueCount,
                                             iowa_ipso_timed_value_t *valueArray);
```

#### Description

`iowa_client_IPSO_update_values()` updates multiple times the value of an IPSO Object sensor.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

The ID of the sensor as returned by `iowa_client_IPSO_add_sensor()`.

##### **valueCount**

The number of values in `valueArray`.

##### **valueArray**

The `iowa_ipso_timed_value_t` list of new values.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`id` does not match any known sensor.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- one of the timestamped value has an negative value.
- the sensor uses a Boolean value and `value` is neither 0.0 nor 1.0.
- the sensor uses a percentage value and `value` is outside the [0.0, 100.0] range.
- the sensor uses a compass direction value and `value` is outside the [0.0, 360.0] range.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation or a call to `iowa_system_gettime()` failed.

#### Header File

`objects/iowa_ipso.h`

#### Notes

`iowa_client_IPSO_update_values()` can only be used when the define `[LWM2M_SUPPORT_TIMESTAMP][LWM2M_SUPPORT_TIMESTAMP]` is set.

The timestamp must be absolute and not relative to the current time, meaning negative values are not accepted. If the timestamp is equal to zero, it is ignored.

Calling `iowa_client_IPSO_update_values()` will overwrite the previous values list. This has multiple consequences:

- If the values have not been sent to the Server, the previous values are lost. Values are only sent if the Server do a Read operation or if the Server has set an Observation.
- If the values are in the way to be sent to the Server and `iowa_client_IPSO_update_values()` is called during the process, some old values will be lost. This API is trying to send the values in best effort. Recent timestamped values are processed in priority before the oldest ones.

Timestamp information is only present if the used Content Format is:

- JSON: `[LWM2M_SUPPORT_JSON][LWM2M_SUPPORT_JSON]`
- SenML JSON: `[LWM2M_SUPPORT_SENML_JSON][LWM2M_SUPPORT_SENML_JSON]`
- SenML CBOR: `[LWM2M_SUPPORT_SENML_CBOR][LWM2M_SUPPORT_SENML_CBOR]`

Evaluation

### 3.17.4 iowa\_client\_IPSO\_remove\_sensor

#### Prototype

```
iowa_status_t iowa_client_IPSO_remove_sensor(iowa_context_t contextP,  
                                              iowa_sensor_t id);
```

#### Description

`iowa_client_IPSO_remove_sensor()` removes from the LwM2M Client an IPSO sensor created with `iowa_client_IPSO_add_sensor()`.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

The ID of the sensor as returned by `iowa_client_IPSO_add_sensor()`.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not an IPSO sensor. Valid `id` are only returned by `iowa_client_IPSO_add_sensor()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

IPSO referred by **id** does not exist.

#### Header File

objects/iowa\_ipso.h

## 3.18 Light Control Object API

This IPSO object is used to control a light source, such as a LED or other light.

To be able to use this object, `iowa_light_control.h` must be included.

### 3.18.1 Callbacks

#### 3.18.1.1 iowa\_light\_control\_update\_state\_callback\_t

This callback is called when the LwM2M Server request the Client to update the state of the light.

```
typedef iowa_status_t (*iowa_light_control_update_state_callback_t)(iowa_sensor_t id,
                                                                    bool powerOn,
                                                                    int dimmer,
                                                                    char *colour,
                                                                    void *userDataCallback
                                                                    ,
                                                                    iowa_context_t
                                                                    contextP);
```

##### **id**

ID of the object

##### **powerOn**

Light power

##### **dimmer**

Dimmer settings

##### **colour**

A string representing a value in some color space

##### **userDataCallback**

Application specific data from `iowa_client_add_light_control_object`. Can be nil.

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

**Return Value** IOWA\_COAP\_NO\_ERROR in case of success or an error status.



## 3.18.2 API

### 3.18.2.1 iowa\_client\_add\_light\_control\_object

#### Prototype

```
iowa_status_t iowa_client_add_light_control_object(
    iowa_context_t context,
    uint16_t optFlags,
    const float powerFactor,
    const char *colorSpace,
    iowa_light_control_update_state_callback_t updateStateCallback,
    void *userDataCallback,
    iowa_sensor_t *idP
);
```

**Description** `iowa_client_add_light_control_object()` creates a light control object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to add optional resources.

##### **powerFactor**

The power factor of the light.

##### **colorSpace**

Color space of the light.

##### **updateStateCallback**

Called to update state of the light. This is called when the server request a new state.

##### **userDataCallback**

Application specific data pass to the callback. Can be nil.

##### **applicationType**

The application type

##### **idP**

Used to store the ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

no update state callback provided means *updateStateCallback* is nil.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_light_control.h`

**Notes** Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- `IOWA_LIGHT_CONTROL_RSC_DIMMER`

- IOWA\_LIGHT\_CONTROL\_RSC\_ON\_TIME
- IOWA\_LIGHT\_CONTROL\_RSC\_CUMULATIVE\_ACTIVE\_POWER
- IOWA\_LIGHT\_CONTROL\_RSC\_POWER\_FACTOR

Moreover, you can add several optional resources at one time by using the following flag:

- IOWA\_LIGHT\_CONTROL\_POWER

The argument *colorSpace* must reflect the color representation of the light. Find below a non-exhaustive list of color spaces:

- RGB
- sRGB
- CMYK
- ...

Evaluation

### 3.18.2.2 iowa\_client\_remove\_light\_control\_object

#### Prototype

```
iowa_status_t iowa_client_remove_light_control_object(iowa_context_t context,
                                                    iowa_sensor_t id);
```

**Description** `iowa_client_remove_light_control_object()` removes a light control object.

#### Arguments

##### **context**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a light control object. Valid `id` are only returned by `iowa_client_add_light_control_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

light control referred by `id` does not exist.

**Header File** `objects/iowa_light_control.h`

### 3.18.2.3 iowa\_client\_light\_control\_set\_state

#### Prototype

```
iowa_status_t iowa_client_light_control_set_state(iowa_context_t context,
                                                iowa_sensor_t id,
                                                bool powerOn,
                                                int dimmer,
                                                const char *colour);
```

**Description** `iowa_client_light_control_set_state()` updates values of a light control object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

##### **powerOn**

Light power

##### **dimmer**

Dimmer settings

##### **colour**

A string representing a value in some color space

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a light control object. Valid `id` are only returned by `iowa_client_add_light_control_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

light control referred by `id` does not exist.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

cannot affect the color. Color space has not been provided. To reconfigure the light control object, delete then readd the object or just add a new one with `iowa_client_add_light_control_object()` and `iowa_client_remove_light_control_object()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_light_control.h`

**Notes** The Light Control Object may contain an “On Time” resource and/or a “Cumulative Active Power” resource. Calling this API updates their respective values.

If `colour` is nil, the value of the resource Colour is not updated.

## 3.19 Location Object API

This LwM2M Object contains information on the device position and speed.

To be able to use this object, `iowa_location.h` must be included.

### 3.19.1 iowa\_client\_add\_location\_object

#### Prototype

```
iowa_status_t iowa_client_add_location_object(iowa_context_t context,
                                             uint16_t optFlags,
                                             iowa_sensor_t *idP);
```

#### Description

`iowa_client_add_location_object()` creates a location object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to add optional resources.

##### **idP**

Used to store the ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_409\_CONFLICT**

a location object already exists. To reconfigure the location object, first delete the object with `iowa_client_remove_location_object()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`objects/iowa_location.h`

#### Notes

Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- IOWA\_LOCATION\_RSC\_ALTITUDE
- IOWA\_LOCATION\_RSC\_RADIUS
- IOWA\_LOCATION\_RSC\_VELOCITY
- IOWA\_LOCATION\_RSC\_SPEED

### 3.19.2 iowa\_client\_remove\_location\_object

#### Prototype

```
iowa_status_t iowa_client_remove_location_object(iowa_context_t context,
                                                iowa_sensor_t id);
```

#### Description

`iowa_client_remove_location_object()` removes a location object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a location object. Valid `id` are only returned by `iowa_client_add_location_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

no location object to remove. `iowa_client_add_location_object()` was not called before, or failed.

#### Header File

`objects/iowa_location.h`

### 3.19.3 iowa\_client\_location\_update

#### Prototype

```
iowa_status_t iowa_client_location_update(iowa_context_t context,
                                          iowa_sensor_t id,
                                          float latitude,
                                          float longitude);
```

#### Description

`iowa_client_location_update()` updates values of a location object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

##### **latitude**

New latitude

##### **longitude**

New longitude

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a location object. Valid `id` are only returned by `iowa_client_add_location_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

object has not been created. First call `iowa_client_add_location_object()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

`iowa_system_gettime()` returned an error.

#### Header File

objects/iowa\_location.h

### 3.19.4 iowa\_client\_location\_update\_full

#### Prototype

```
iowa_status_t iowa_client_location_update_full(iowa_context_t context,
                                              iowa_sensor_t id,
                                              float latitude,
                                              float longitude,
                                              float altitude,
                                              float radius,
                                              size_t velocityLength,
                                              uint8_t* velocity,
                                              float speed);
```

#### Description

`iowa_client_location_update_full()` updates values of a location object. Optional resources are included.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

##### **latitude**

New latitude

##### **longitude**

New longitude

##### **altitude**

New altitude

##### **radius**

Indicates the size in meters of a circular area around a point of geometry.

##### **velocityLength**

Length of the velocity array

##### **velocity**

The velocity of the device as defined in 3GPP 23.032 GAD specification.

##### **speed**

Speed is the time rate of change in position of a LwM2M Client without regard for direction: the scalar component of velocity.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a location object. Valid `id` are only returned by `iowa_client_add_location_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

object has not been created. First call `iowa_client_add_location_object()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.



- `iowa_system_gettime()` returned an error.

### Header File

`objects/iowa_location.h`

Evaluation

## 3.20 Magnetometer Object API

This IPSO object can be used to represent a 1-3 axis magnetometer with optional compass direction.

To be able to use this object, `iowa_magnetometer.h` must be included.

### 3.20.1 iowa\_client\_add\_magnetometer\_object

#### Prototype

```
iowa_status_t iowa_client_add_magnetometer_object(iowa_context_t context,
                                                  uint16_t optFlags,
                                                  const char *sensorUnits,
                                                  iowa_sensor_t *idP);
```

#### Description

`iowa_client_add_magnetometer_object()` creates a magnetometer object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to add optional resources.

##### **sensorUnits**

Measurement units definition.

##### **idP**

Used to store the ID of the object.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`objects/iowa_magnetometer.h`

#### Notes

Please refer to the [OMA LightweightM2M \(LwM2M\) Object and Resource Registry](#) to see how the object is defined: resources, resources type, ...

When no optional flags are provided only mandatory resources of the object are implemented.

To add optional resources, you can use the following flags:

- IOWA\_MAGNETOMETER\_RSC\_Y\_VALUE
- IOWA\_MAGNETOMETER\_RSC\_Z\_VALUE
- IOWA\_MAGNETOMETER\_RSC\_COMPASS\_DIRECTION

Moreover, you can add several optional resources at one time by using the following flag:

- IOWA\_MAGNETOMETER\_3\_AXIS

### 3.20.2 iowa\_client\_remove\_magnetometer\_object

#### Prototype

```
iowa_status_t iowa_client_remove_magnetometer_object(iowa_context_t context,
                                                    iowa_sensor_t id);
```

#### Description

`iowa_client_remove_magnetometer_object()` removes a magnetometer object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a magnetometer object. Valid `id` are only returned by `iowa_client_add_magnetometer_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

magnetometer referred by **id** does not exist.

#### Header File

`objects/iowa_magnetometer.h`

### 3.20.3 iowa\_client\_magnetometer\_update\_values

#### Prototype

```
iowa_status_t iowa_client_magnetometer_update_values(iowa_context_t context,
                                                    iowa_sensor_t id,
                                                    float xValue,
                                                    float yValue,
                                                    float zValue,
                                                    float compassDirection);
```

#### Description

`iowa_client_magnetometer_update_values()` updates values of a magnetometer object.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the object

##### **xValue**

X value axis

##### **yValue**

Y value axis

##### **zValue**

Z value axis

##### **compassDirection**

Measured Direction.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a magnetometer object. Valid `id` are only returned by `iowa_client_add_magnetometer_object()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

magnetometer referred by `id` does not exist.

#### Header File

objects/iowa\_magnetometer.h

## 3.21 Software Component Object API

This LwM2M object provides the resources needed to activate/deactivate software components on the device.

To be able to use this object, `iowa_software_component.h` must be included and **IOWA\_SUPPORT\_SOFTWARE\_COMPONENT\_OBJECT** must be defined before building IOWA.

### 3.21.1 Data Structures and Constants

#### 3.21.1.1 iowa\_sw\_cmp\_info\_t

This structure contains the description of a software component's information which could be set by users.

```
typedef struct
{
    const char    *identityP;
    const uint8_t *packP;
    size_t        packLength;
    const char    *versionP;
} iowa_sw_cmp_info_t;
```

##### **identityP**

Name or identifier of the software component, with size < 255. This can be nil.

##### **packP**

Link to opaque data describing the software component. This can be nil.

##### **packLength**

Length in bytes of the opaque data pointed by *packP*.

##### **versionP**

Version of the software component, with size < 255. This can be nil.

**Note** This structure will at least provide an identity (*identityP*) or a pack (*packP*) to identify the component.

### 3.21.2 Callbacks

#### 3.21.2.1 iowa\_sw\_cmp\_update\_callback\_t

This is the update callback, called when the Server adds or removes the software components.

```
typedef iowa_status_t(*iowa_sw_cmp_update_callback_t)(iowa_sensor_t id,
                                                    iowa_dm_operation_t operation,
                                                    iowa_sw_cmp_info_t *infoP,
                                                    bool activationState,
                                                    void *userDataP,
                                                    iowa_context_t contextP);
```

**id**

ID of the corresponding software component.

**operation**

the operation performed by the Server on this software component (either **IOWA\_DM\_CREATE** or **IOWA\_DM\_DELETE**).

**infoP**

software component information.

**activationState**

initial activation state. Should be ignored if no `iowa_sw_cmp_activation_callback_t()` was passed to `iowa_client_enable_software_component()`.

**userDataP**

The data passed to `iowa_client_enable_software_package_management()`.

**contextP**

The IOWA context on which `iowa_client_enable_software_package_management()` was called.

**Return Value** IOWA\_COAP\_NO\_ERROR in case of success or an error status.

#### 3.21.2.2 iowa\_sw\_cmp\_activation\_callback\_t

This is the activation callback, called when the Server requests the device to activate or deactivate a software component.

```
typedef iowa_status_t(*iowa_sw_cmp_activation_callback_t) (iowa_sensor_t id,
                                                         bool activationState,
                                                         void *userDataP,
                                                         iowa_context_t contextP);
```

**id**

ID of the corresponding software component.

**activationState**

activation state requested.

**userDataP**

The data passed to `iowa_client_enable_software_component()`.

**contextP**

The IOWA context on which `iowa_client_enable_software_component()` was called.

**Return Value** IOWA\_COAP\_NO\_ERROR in case of success or an error status.

### 3.21.3 API

#### 3.21.3.1 iowa\_client\_enable\_software\_component

##### Prototype

```
iowa_status_t iowa_client_enable_software_component(
    iowa_context_t contextP,
    iowa_sw_cmp_update_callback_t updateCb,
    iowa_sw_cmp_activation_callback_t activationCb,
    void *userDataP
);
```

**Description** `iowa_client_enable_software_component()` enables the software component feature.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **updateCb**

The update callback called when the Server adds or removes a software component. This can be nil.

###### **activateCb**

The activate callback, called when the Server requests the device to activate or deactivate a software component. This can be nil.

###### **userDataP**

Passed as argument to the callback. This can be nil.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_409\_CONFLICT**

Software component feature is already configured. To reconfigure the software component, disable it before with `iowa_client_disable_software_component()`.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_software_component.h`

### 3.21.3.2 iowa\_client\_disable\_software\_component

#### Prototype

```
iowa_status_t iowa_client_disable_software_component(iowa_context_t contextP);
```

**Description** `iowa_client_disable_software_component()` disables the software component feature.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

Software component feature was not enabled. `iowa_client_enable_software_component()` was not called before, or failed.

**Header File** `objects/iowa_software_component.h`



### 3.21.3.3 iowa\_client\_add\_software\_component

#### Prototype

```
iowa_status_t iowa_client_add_software_component(iowa_context_t contextP,
                                                iowa_sw_cmp_info_t *infoP,
                                                bool activationState,
                                                iowa_sensor_t *idP);
```

**Description** `iowa_client_add_software_component()` adds a software component.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **infoP**

software component information.

##### **activationState**

current activation state of the software component. Ignored if no `iowa_sw_cmp_activation_callback_t()` was passed to `iowa_client_enable_software_component()`.

##### **idP**

Used to store the ID of the created software component. Not checked at runtime.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- *infoP* is nil.
- both *infoP::identityP* and *infoP::packP* are nil.
- a string in *infoP* is longer than 255 characters.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

Software component feature was not enabled. Call first `iowa_client_enable_software_component()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_software_component.h`

**Notes** *infoP* must provide at least an identity (*infoP::identityP*) or a pack (*infoP::packP*) to identify the component.

The “const” elements pointed by the fields of *infoP* are not duplicated nor freed by IOWA. Make sure they are available until corresponding `iowa_client_remove_software_component()`, `iowa_client_disable_software_component()`, or `iowa_close()` is called.

### 3.21.3.4 iowa\_client\_remove\_software\_component

#### Prototype

```
iowa_status_t iowa_client_remove_software_component(iowa_context_t contextP,  
                                                    iowa_sensor_t id);
```

**Description** `iowa_client_remove_software_component()` removes a software component.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the corresponding software component.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a software component. Valid `id` are only returned by `iowa_client_add_software_component()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

software component referred by **id** does not exist.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

Software component feature was not enabled. Call first `iowa_client_enable_software_component()`.

**Header File** `objects/iowa_software_component.h`

### 3.21.3.5 iowa\_client\_software\_component\_update\_state

#### Prototype

```
iowa_status_t iowa_client_software_component_update_state(iowa_context_t contextP,  
                                                         iowa_sensor_t id,  
                                                         bool activationState);
```

**Description** `iowa_client_software_component_update_state()` updates a software component's activation state.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the corresponding software component.

##### **activationState**

New activation state of the software component.

**Note** This API has no effect if no `iowa_sw_cmp_activation_callback_t()` was passed to `iowa_client_enable_software_component()` since the Activation State resource is not presented to the LwM2M Server.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`id` does not match any known software component.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

Software component feature was not enabled. Call first `iowa_client_enable_software_component()`.

**Header File** `objects/iowa_software_component.h`

## 3.22 Software Management Object API

In LightweightM2M, the Software Management mechanism is used to install and activate software components to a LwM2M Client.

To be able to use this object, `iowa_software_management.h` must be included and **IOWA\_SUPPORT\_SOFTWARE\_MANAGEMENT\_OBJECT** must be defined before building the library.

The [Device Update][Device Update]/[Software Management][Software Management] part of this specification adds more explanation about its mechanism and how to use it with IOWA.

**Note:** As there is currently some confusion on the layout of the Software Management Object, IOWA uses the definition provided in [LwM2M Overview][LwM2M Overview]/[Software Management Object][Software Management Object].

### 3.22.1 Data Structures and Constants

#### 3.22.1.1 iowa\_sw\_pkg\_result\_t

This enumeration is used to update the operation result from `iowa_sw_pkg_download_callback_t()`, `iowa_sw_pkg_write_callback_t()` and `iowa_sw_pkg_install_callback_t()` callbacks. It has the following values:

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_SUCCESSFUL**

success of any operation made on the software package (verification, installation, uninstallation, activation, deactivation)

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_DOWNLOADING\_SUCCESSFUL**

success of the new software package download. (*downloadCb* only)

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_OUT\_OF\_STORAGE**

not enough storage for the new software package. (*downloadCb* or *writeCb* only)

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_OUT\_OF\_MEMORY**

out of memory error during the download of the new software package. (*downloadCb* or *writeCb* only)

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_CONNECTION\_LOST**

connection lost during the download of the new software package. (*downloadCb* or *writeCb* only)

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_INTEGRITY\_CHECK\_FAILURE**

integrity check failure of the new software package. (*downloadCb* or *writeCb* only)

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_UNSUPPORTED\_TYPE**

unsupported new software package type.

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_INVALID\_URI**

invalid URI to download the new software package. (*downloadCb* only)

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_UPDATE\_FAILED**

device defined update error.

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_INSTALLED\_FAILURE**

new software installation failure. (*installCb* only)

**IOWA\_SW\_PKG\_UPDATE\_RESULT\_UNINSTALLED\_FAILURE**

software uninstallation failure. (*installCb* only)

#### 3.22.1.2 iowa\_sw\_pkg\_state\_t

This enumeration is used to control current state in `iowa_client_add_software_package()` and `iowa_client_software_package_update_state()`. It has the following values:

**IOWA\_SW\_PKG\_STATE\_UNINSTALLED**

software is uninstalled. (default value)

**IOWA\_SW\_PKG\_STATE\_INSTALLED**

software is installed.

## **IOWA\_SW\_PKG\_STATE\_ACTIVATED**

software is activate. Useful only if Software components are linked, otherwise same behavior than installed.

### **3.22.1.3 iowa\_sw\_pkg\_write\_cmd\_t**

This enumeration is used in `iowa_sw_pkg_write_callback_t()` callback. It has the following values:

#### **IOWA\_SW\_PKG\_COMMAND\_RESET**

To start software package packet writing.

#### **IOWA\_SW\_PKG\_COMMAND\_WRITE**

To indicate other software package piece of the complete packet.

### **3.22.1.4 iowa\_sw\_pkg\_install\_cmd\_t**

This enumeration is used in `iowa_sw_pkg_install_callback_t()` callback. It has the following values:

#### **IOWA\_SW\_PKG\_COMMAND\_INSTALL**

software installation is requested.

#### **IOWA\_SW\_PKG\_COMMAND\_UNINSTALL**

software uninstallation is requested.

#### **IOWA\_SW\_PKG\_COMMAND\_PREPARE\_FOR\_UPDATE**

software uninstallation is requested to prepare an update.

### **3.22.1.5 iowa\_sw\_pkg\_optional\_info\_t**

This structure contains the description of a optional software package's information which could be set by users.

```
typedef struct
{
    iowa_sensor_t    *swComponentLinkP;
    uint16_t         swComponentLinkCount;
} iowa_sw_pkg_optional_info_t;
```

#### **swComponentLinkP**

Software Components downloaded and installed in scope of the present SW Update Package. This can be nil. Each `swComponentLinkP` sensor id must have been provided by [Software Component Object APIs](#).

#### **swComponentLinkCount**

Software Components Link count.

## 3.22.2 Callbacks

### 3.22.2.1 iowa\_sw\_pkg\_update\_callback\_t

This is the update callback, called when the Server adds or removes the software packages.

```
typedef iowa_status_t(*iowa_sw_pkg_update_callback_t) (iowa_sensor_t id,
                                                    iowa_dm_operation_t operation,
                                                    const char *pkgNameP,
                                                    const char *pkgVersionP,
                                                    iowa_sw_pkg_optional_info_t *optP,
                                                    void *userDataP,
                                                    iowa_context_t contextP);
```

#### **id**

ID of the corresponding software package.

#### **operation**

the operation performed by the Server on this software package (either IOWA\_DM\_CREATE or IOWA\_DM\_DELETE).

#### **pkgNameP**

Name of the software package.

#### **pkgVersionP**

Version of the software package.

#### **optP**

Optional information. This can be nil.

#### **userDataP**

The data passed to `iowa_client_enable_software_package_management()`.

#### **contextP**

The IOWA context on which `iowa_client_enable_software_package_management()` was called.

**Return Value** IOWA\_COAP\_NO\_ERROR in case of success or an error status.

### 3.22.2.2 iowa\_sw\_pkg\_download\_callback\_t

This is the download callback, called when the Server requests the device to download a new software Package (new value in "Package URI").

When the packet is downloaded, users should call `iowa_client_set_software_package_command_result()` with IOWA\_SW\_PKG\_UPDATE\_RESULT\_DOWNLOADING\_SUCCESSFUL result if successful or an error result otherwise.

When the packet is verified, users should call `iowa_client_set_software_package_command_result()` with IOWA\_SW\_PKG\_UPDATE\_RESULT\_SUCCESSFUL result if successful or an error result otherwise.

```
typedef void(*iowa_sw_pkg_download_callback_t) (iowa_sensor_t id,
                                                const char *uriP,
                                                const char *userNameP,
                                                const char *passwordP,
                                                void *userDataP,
                                                iowa_context_t contextP);
```

#### **id**

ID of the corresponding software package instance.

#### **uriP**

URI to download the package from.

#### **userNameP**

User Name for access to SW Update Package in pull mode, with size < 255. Key based mechanism can alternatively use for talking to the component server instead of user name and password combination. This can be nil.

### **passwordP**

Password for access to SW Update Package in pull mode, with size < 255. This can be nil.

### **userDataP**

The data passed to `iowa_client_enable_software_package_management()`.

### **contextP**

The IOWA context on which `iowa_client_enable_software_package_management()` was called.

### **3.22.2.3 iowa\_sw\_pkg\_write\_callback\_t**

This is the write callback, called several times when the Server pushes the new software package to the device (new value in "Package"). The expected behavior is the same as writing to a file stream i.e. unless it is reset, written data are appended to the previous ones.

```
typedef iowa_sw_pkg_result_t(*iowa_sw_pkg_write_callback_t) (iowa_sensor_t id,
                                                             iowa_sw_pkg_write_cmd_t cmd,
                                                             size_t dataLength,
                                                             uint8_t *dataP,
                                                             void *userDataP,
                                                             iowa_context_t contextP);
```

At the start of the push of the software package or if the LwM2M Server cancels it, this callback is called with the following parameters:

#### **id**

ID of the corresponding software package instance.

#### **cmd**

**IOWA\_SW\_PKG\_COMMAND\_RESET**

#### **dataLength**

0

#### **dataP**

NULL

#### **userDataP**

The data passed to `iowa_client_enable_software_package_management()`.

#### **contextP**

The IOWA context on which `iowa_client_enable_software_package_management()` was called.

When the software package is received, this callback is called several times with the following parameters:

#### **id**

ID of the corresponding software package instance.

#### **cmd**

**IOWA\_SW\_PKG\_COMMAND\_WRITE**

#### **dataLength**

The length of the buffer pointed by `dataP`.

#### **data**

The next chunk of the software package to write.

#### **userDataP**

The data passed to `iowa_client_enable_software_package_management()`.

#### **contextP**

The IOWA context on which `iowa_client_enable_software_package_management()` was called.

At the end of the push of the software package, this callback is called with the following parameters:

#### **id**

ID of the corresponding software package instance.

**cmd**

IOWA\_SW\_MGMT\_PACKAGE\_WRITE

**dataLength**

0

**data**

NULL

**userDataP**

The data passed to `iowa_client_enable_software_package_management()`.

**contextP**

The IOWA context on which `iowa_client_enable_software_package_management()` was called.

**Return Value** IOWA\_SW\_PKG\_UPDATE\_RESULT\_SUCCESSFUL in case of success or an error status.

### 3.22.2.4 iowa\_sw\_pkg\_install\_callback\_t

This is the install callback, called when the Server requests the device to install or uninstall the software Package.

When the installation finishes, users should call `iowa_client_set_software_package_command_result()` with IOWA\_SW\_PKG\_UPDATE\_RESULT\_SUCCESSFUL result if successful or an error result otherwise.

```
typedef void(*iowa_sw_pkg_install_callback_t) (iowa_sensor_t id,
                                              iowa_sw_pkg_install_cmd_t cmd,
                                              void *userDataP,
                                              iowa_context_t contextP);
```

**id**

ID of the corresponding software package instance.

**cmd**

installed state requested. See `iowa_sw_pkg_install_cmd_t`.

**userDataP**

The data passed to `iowa_client_enable_software_package_management()`.

**contextP**

The IOWA context on which `iowa_client_enable_software_package_management()` was called.



### 3.22.3 API

#### 3.22.3.1 iowa\_client\_enable\_software\_package\_management

##### Prototype

```
iowa_status_t iowa_client_enable_software_package_management(
    iowa_context_t contextP,
    iowa_sw_pkg_update_callback_t updateCb,
    iowa_sw_pkg_download_callback_t downloadCb,
    iowa_sw_pkg_write_callback_t writeCb,
    iowa_sw_pkg_install_callback_t installCb,
    void *userDataP
);
```

**Description** `iowa_client_enable_software_package_management()` enables the software package management feature.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **updateCb**

The update callback called when the Server adds or removes the software packages. This can be nil.

###### **downloadCb**

The download callback, called when the Server requests the device to download a new software Package (new value in "Package URI"). This can be nil.

###### **writeCb**

The write callback, called several times when the Server pushes the new software Package to the device (new value in "Package"). This can be nil.

###### **installCb**

The install callback, called when the Server requests the device to install or uninstall the software Package.

###### **userDataP**

Passed as argument to the callbacks. This can be nil.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **COAP\_400\_BAD\_REQUEST**

either:

- `installCb` is nil.
- both `downloadCb` and `writeCb` are nil. At least one must be defined.

###### **IOWA\_COAP\_409\_CONFLICT**

Software package feature is already configured. To reconfigure the software package, disable it before with `iowa_client_disable_software_package_management()`.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_software_management.h`

**Notes** The LwM2M Server has two methods to provide the software package:

- the “pull” method: the LwM2M Server provides the URI of the software package and the LwM2M Client downloads it directly. To use this method, *downloadCb* must be set.
- the “push” method: the LwM2M Server writes the software package in a LwM2M Resource exposed by the Client. To use this method, *writeCb* must be set.

The Client can support both methods at the same time and must at least provide one of them.

*downloadCb* and *updateCb* do not return any value. The progress and result of their operation are indicated asynchronously by calling `iowa_client_set_software_package_command_result()`.

Evaluation

### 3.22.3.2 iowa\_client\_disable\_software\_package\_management

#### Prototype

```
iowa_status_t iowa_client_disable_software_package_management(iowa_context_t contextP);
```

**Description** `iowa_client_disable_software_package_management()` disables the software package management feature.

#### Arguments

##### *contextP*

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

Software package feature was not enabled. `iowa_client_enable_software_package_management()` was not called before, or failed.

**Header File** `objects/iowa_software_management.h`

### 3.22.3.3 iowa\_client\_add\_software\_package

#### Prototype

```
iowa_status_t iowa_client_add_software_package(iowa_context_t contextP,
                                              const char *pkgNameP,
                                              const char *pkgVersionP,
                                              iowa_sw_pkg_state_t state,
                                              iowa_sw_pkg_optional_info_t *optP,
                                              iowa_sensor_t *idP);
```

**Description** `iowa_client_add_software_package()` adds a software package instance.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **pkgNameP**

Name of the software package.

##### **pkgVersionP**

Version of the software package.

##### **state**

State of the software package. (default value: `IOWA_SW_PKG_STATE_UNINSTALLED`)

##### **optP**

Optional information. This can be nil.

##### **idP**

Used to store the ID of the created software package instance. Not checked at runtime.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- `pkgNameP` is nil.
- `pkgVersionP` is nil.
- `optP` has invalid format.
- Any string is larger than 255 characters.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

Software package feature was not enabled. Call first `iowa_client_enable_software_package_management()`.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_software_management.h`

### 3.22.3.4 iowa\_client\_remove\_software\_package

#### Prototype

```
iowa_status_t iowa_client_remove_software_package(iowa_context_t contextP,  
                                                  iowa_sensor_t id);
```

**Description** `iowa_client_remove_software_package()` removes a software package instance.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the corresponding software package instance.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`id` is not a software package. Valid `id` are only returned by `iowa_client_add_software_package()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

software package referred by `id` does not exist.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

Software package feature was not enabled. Call first `iowa_client_enable_software_package_management()`.

**Header File** `objects/iowa_software_management.h`

### 3.22.3.5 iowa\_client\_software\_package\_update\_state

#### Prototype

```
iowa_status_t iowa_client_software_package_update_state(iowa_context_t contextP,  
                                                         iowa_sensor_t id,  
                                                         iowa_sw_pkg_state_t state);
```

**Description** `iowa_client_software_package_update_state()` updates a software package instance's state.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the corresponding software package instance.

##### **state**

state of the software package.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`id` does not match any known software package.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

Software package feature was not enabled. Call first `iowa_client_enable_software_package_management()`.

**Header File** `objects/iowa_software_management.h`

### 3.22.3.6 iowa\_client\_set\_software\_package\_command\_result

#### Prototype

```
iowa_status_t iowa_client_set_software_package_command_result(iowa_context_t contextP,  
                                                             iowa_sensor_t id,  
                                                             iowa_sw_pkg_result_t result)  
                                                             ;
```

**Description** `iowa_client_set_software_package_command_result()` informs the IOWA stack of the result of the callbacks *downloadCb* and *installCb* of `iowa_client_enable_software_package_management()`.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **id**

ID of the corresponding software package instance.

##### **result**

The result of the software package callbacks.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

*id* does not match any known software package.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

Software package feature was not enabled. Call first `iowa_client_enable_software_package_management()`.

**Header File** `objects/iowa_software_management.h`

## 3.23 MQTT Object API

This part allows the possibility to connect with an MQTT Broker.

To be able to use these objects, `iowa_mqtt_objects.h` must be included.

### 3.23.1 Data Structures and Constants

#### 3.23.1.1 iowa\_mqtt\_broker\_t

```
typedef struct
{
    char    *uri;
    char    *clientId;
    bool    cleanSession;
    uint16_t keepAlive;
    char    *userName;
    uint8_t *password;
    size_t  passwordLength;

    iowa_security_mode_t  securityMode;
    iowa_cert_usage_mode_t certificateUsage;
    uint8_t               *identity;
    size_t                identityLength;
    uint8_t               *brokerIdentity;
    size_t                brokerIdentityLength;
    uint8_t               *privateKey;
    size_t                privateKeyLength;
} iowa_mqtt_broker_t;
```

##### **uri**

The URI to reach the MQTT Broker as a nil-terminated string e.g. "tcp://[::1]:1883".

##### **clientId**

MQTT Client Identifier to use when connecting to this MQTT broker.

##### **cleanSession**

A boolean that's indicate to the MQTT broker to create a persistent session.

##### **keepAlive**

The maximum time in seconds that's the client take to send or receive a message.

##### **userName**

The User Name to declare in the MQTT CONNECT message.

##### **password**

The Password value to declare in the MQTT CONNECT message.

##### **passwordLength**

The length of the broker's password.

##### **securityMode**

The security mode to use when connecting to this LwM2M Server. See `[iowa_security_mode_t][iowa_security_mode_t]`.

##### **certificateUsage**

The Certificate Usage Resource specifies the semantic of the certificate or raw public key stored in the "MQTT Broker Public Key" Resource, which is used to match the certificate presented in the TLS/DTLS handshake. See `[iowa_cert_usage_mode_t][iowa_cert_usage_mode_t]`.

When this Resource is absent, value **IOWA\_CERTIFICATE\_USAGE\_DOMAIN\_ISSUED\_CERTIFICATE** for domain issued certificate mode is assumed.

##### **identity**

Stores the Device's certificate, public key (RPK mode) or PSK Identity (PSK mode).



### ***identityLength***

The identity length.

### ***brokerIdentity***

Stores the MQTT Broker's certificate, public key (RPK mode) or trust anchor. The Certificate Usage Resource determines the content of this resource.

### ***brokerIdentityLength***

The length of the broker's Identity.

### ***privateKey***

Stores the secret key (PSK mode) or private key (RPK or certificate mode).

### ***privateKeyLength***

The private key's length

## **3.23.1.2 iowa\_mqtt\_publication\_t**

```
typedef struct
{
    iowa_sensor_t      brokerId;
    char               *source;
    char               *topic;
    uint8_t            qos;
    bool               retain;
    bool               active;
    iowa_content_format_t encoding;
} iowa_mqtt_publication_t;
```

### ***brokerId***

The ID of the broker to be used.

### ***source***

The source of the data to publish (e.g. "(", or";"). If this Resource is empty, the published data are implementation dependent.

### ***topic***

The MQTT topic to publish to.

### ***qos***

The Quality of Service value to use when publishing.

### ***retain***

The RETAIN flag value to use when publishing.

### ***active***

A boolean to indicate if the Resource is not present, the Device publishes the data pointed by the Source Resource to the MQTT Broker pointed by the Broker Resource using the MQTT topic indicated in the Topic Resource. If false, the Device does nothing.

### ***encoding***

A CoAP Content-Format value used to encode the data in the MQTT Publish message. If this Resource is not present or equal to 65535, the encoding of the data is implementation dependent.

### 3.23.2 Callbacks

#### 3.23.2.1 iowa\_mqtt\_broker\_update\_callback\_t

The callback called when a LwM2M Server modifies the MQTT Broker Object.

```
typedef void (*iowa_mqtt_broker_update_callback_t) (iowa_dm_operation_t operation,
                                                    iowa_sensor_t brokerId,
                                                    iowa_mqtt_broker_t *brokerDetailsP,
                                                    void *userData,
                                                    iowa_context_t contextP);
```

**operation**

the type of the operation among IOWA\_DM\_READ, IOWA\_DM\_WRITE and IOWA\_DM\_DELETE.

**brokerId**

the ID of the modified MQTT broker.

**brokerDetailsP**

the details of the modified MQTT broker.

**userData**

Passed as argument to the callbacks. This can be nil.

**contextP**

the IOWA context on which iowa\_client\_enable\_mqtt\_broker() was called.

#### 3.23.2.2 iowa\_mqtt\_publication\_update\_callback\_t

The callback called when a LwM2M Server modifies the MQTT Publication Object.

```
typedef void (*iowa_mqtt_publication_update_callback_t) (iowa_dm_operation_t operation,
                                                         iowa_sensor_t publicationId,
                                                         iowa_mqtt_publication_t *
                                                         publicationDetailsP,
                                                         void *userData,
                                                         iowa_context_t contextP);
```

**operation**

the type of the operation among IOWA\_DM\_READ and IOWA\_DM\_WRITE.

**publicationId**

the ID of the modified MQTT Publication.

**publicationDetailsP**

the details of the modified MQTT Publication.

**userData**

Passed as argument to the callbacks. This can be nil.

**contextP**

the IOWA context on which iowa\_client\_enable\_mqtt\_publication() was called.

### 3.23.3 API

#### 3.23.3.1 iowa\_client\_enable\_mqtt\_broker

##### Prototype

```
iowa_status_t iowa_client_enable_mqtt_broker(iowa_context_t contextP,  
                                             iowa_mqtt_broker_update_callback_t brokerCb,  
                                             void * userData);
```

**Description** `iowa_client_enable_mqtt_broker()` enables the MQTT brokers management.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **brokerCb**

The broker callback called when a LwM2M Server modify the MQTT brokers.

###### **userDataP**

Passed as argument to the callbacks. This can be nil.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

`brokerCb` is nil.

###### **IOWA\_COAP\_409\_CONFLICT**

MQTT brokers management is already enabled.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_mqtt_objects.h`

### 3.23.3.2 iowa\_client\_disable\_mqtt\_broker

#### Prototype

```
iowa_status_t iowa_client_disable_mqtt_broker(iowa_context_t contextP);
```

**Description** `iowa_client_disable_mqtt_broker()` disables the MQTT brokers management.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

MQTT brokers management is not enabled.

**Header File** `objects/iowa_mqtt_objects.h`

### 3.23.3.3 iowa\_client\_add\_mqtt\_broker

#### Prototype

```
iowa_status_t iowa_client_add_mqtt_broker(iowa_context_t contextP,
                                         uint16_t optFlags,
                                         const iowa_mqtt_broker_t * brokerDetailsP,
                                         iowa_sensor_t * brokerIdP);
```

**Description** `iowa_client_add_mqtt_broker()` adds an MQTT broker instance.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to add optional resources.

##### **brokerDetailsP**

The details of the MQTT Broker. Copied internally by IOWA.

##### **brokerIdP**

Used to store the ID of the created MQTT Broker instance.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- Invalid `brokerDetailsP`.
- `brokerIdP` is nil.

**IOWA\_COAP\_412\_PRECONDITION\_FAILED** MQTT brokers management is not enabled. Call `iowa_client_enable_mqtt_broker()` first.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation or a call to `iowa_system_gettime()` failed.

**Header File** `objects/iowa_mqtt_objects.h`

#### Notes

##### **The invalid broker's details are**

either:

- The client identity is nil.
- The password length is 0 and the password is not nil.
- The identity length is 0 and the identity is not nil.
- The private key length is 0 and the private key is not nil.
- The broker's identity length is 0 and the broker's identity is not nil.
- The security mode is `IOWA_SEC_NONE` and the identity length and/or private key length and/or broker's identity length are greater than 0.
- The security mode is different than `IOWA_SEC_NONE` and the identity length and/or private key length equal to 0.

- The security mode is IOWA\_SEC\_RAW\_PUBLIC\_KEY or IOWA\_SEC\_CERTIFICATE and the broker identity length equal to 0.
- The security mode value is unknown.
- The certificate usage value is unknown.

To add optional resource, you can use the following flag:

- IOWA\_MQTT\_BROKER\_CERTIFICATE\_USAGE : a flag to set the broker certificate usage resource, if this resource is not set by the user, domain issued certificate mode is assumed.

Evaluation

### 3.23.3.4 iowa\_client\_remove\_mqtt\_broker

#### Prototype

```
iowa_status_t iowa_client_remove_mqtt_broker(iowa_context_t contextP,  
                                              iowa_sensor_t brokerId);
```

**Description** `iowa_client_remove_mqtt_broker()` removes an MQTT broker instance.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **brokerId**

The ID assigned to the MQTT Broker by IOWA.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

*brokerId* is not valid. Valid *brokerId* are only returned by `iowa_client_add_mqtt_broker()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

*brokerId* does not match a known MQTT broker.

**IOWA\_COAP\_412\_PRECONDITION\_FAILED** MQTT brokers management is not enabled. Call `iowa_client_enable_mqtt_broker()` first.

**Header File** `objects/iowa_mqtt_objects.h`

### 3.23.3.5 iowa\_client\_get\_mqtt\_broker

#### Prototype

```
iowa_mqtt_broker_t * iowa_client_get_mqtt_broker(iowa_context_t contextP,  
                                                iowa_sensor_t brokerId);
```

**Description** `iowa_client_get_mqtt_broker()` retrieves the details of an MQTT broker.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **brokerId**

The ID assigned to the MQTT Broker by IOWA.

**Return Value** The MQTT broker's details, or NULL if the MQTT brokers management is not enabled or if *brokerId* does not match a known MQTT broker.

**Notes** The returned pointer points to the internal data of IOWA and not to duplicated information. It is advised to not modify it.

**Header File** `objects/iowa_mqtt_objects.h`



### 3.23.3.6 iowa\_client\_enable\_mqtt\_publication

#### Prototype

```
iowa_status_t iowa_client_enable_mqtt_publication(iowa_context_t contextP,  
                                                  iowa_mqtt_publication_update_callback_t  
                                                  publicationCB,  
                                                  void *userData);
```

**Description** `iowa_client_enable_mqtt_publication()` enables the MQTT Publication management.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **publicationCB**

The broker callback called when a LwM2M Server modify the MQTT brokers.

##### **userDataP**

Passed as argument to the callbacks. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

`publicationCB` is nil.

##### **IOWA\_COAP\_409\_CONFLICT**

MQTT Publication Object already exists.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

**Header File** `objects/iowa_mqtt_objects.h`

### 3.23.3.7 iowa\_client\_disable\_mqtt\_publication

#### Prototype

```
iowa_status_t iowa_client_disable_mqtt_publication(iowa_context_t contextP);
```

**Description** `iowa_client_disable_mqtt_publication()` disables the MQTT Publication management.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

MQTT publication Object not found.

**Header File** `objects/iowa_mqtt_objects.h`

### 3.23.3.8 iowa\_client\_add\_mqtt\_publication

#### Prototype

```
iowa_status_t iowa_client_add_mqtt_publication(iowa_context_t contextP,
                                              uint16_t optFlags,
                                              const iowa_mqtt_publication_t *
                                              publicationDetailsP,
                                              iowa_sensor_t *publicationIdP);
```

**Description** `iowa_client_add_mqtt_publication()` adds an MQTT publication instance.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **optFlags**

Optional flags to indicate optional resources.

##### **publicationDetailsP**

publicationDetailsP: details of the MQTT Publication. Copied internally by IOWA.

##### **publicationIdP**

Used to store the ID of the created MQTT Publication instance.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- Invalid `publicationDetailsP`.
- `publicationIdP` is nil.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

MQTT publication management was not enabled. Call `iowa_client_enable_mqtt_publication()` first.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation or a call to `iowa_system_gettime()` failed.

Invalid publication source format.

**Header File** `objects/iowa_mqtt_objects.h`

#### Notes

**The invalid publication's details are either:**

- The publication's topic is nil.

- The publication's source is nil.
- The publication's QOS is greater than 2.

To add optional resources, you can use the following flag:

- `IOWA_MQTT_PUBLICATION_RSC_ENCODING` : a flag to set the publication encoding resource, if this resource is not set by the user, the encoding of the data is implementation dependent.
- `IOWA_MQTT_PUBLICATION_RSC_ACTIVE` : a flag to set the publication active resource. if this resource is not set by the user, it will be assumed to be true.

### 3.23.3.9 iowa\_client\_remove\_mqtt\_publication

#### Prototype

```
iowa_status_t iowa_client_remove_mqtt_publication(iowa_context_t contextP,  
                                                  iowa_sensor_t publicationId);
```

**Description** `iowa_client_remove_mqtt_publication()` removes an MQTT Publication instance.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **publicationId**

The ID assigned to the MQTT Publication by IOWA.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

*publicationId* is not an MQTT object. Valid *publicationId* are only returned by `iowa_client_add_mqtt_publication()`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

MQTT object referred by **publicationId** does not exist.

##### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

MQTT publication management was not enabled. Call `iowa_client_enable_mqtt_publication()` first.

**Header File** `objects/iowa_mqtt_objects.h`

## 4 | Server Mode API Reference

The functions explained below are defined inside the file *include/iowa\_server.h*.

### 4.1 Server pseudo code

```
#include "iowa_server.h"

int main(int argc,
        char *argv[])
{
    iowa_context_t iowaH;
    iowa_status_t result;
    int serverSocket;

    /*****
    * Initialization
    */

    serverSocket = open_udp_socket();

    iowaH = iowa_init(NULL);

    result = iowa_server_configure(iowaH, client_monitor, NULL, NULL);

    /*****
    * "Main loop"
    */

    while (result == IOWA_COAP_NO_ERROR)
    {
        result = iowa_step(iowaH, 10);

        if (isDataAvailable(serverSocket))
        {
            void * newConnection;

            newConnection = create_new_connection();
            result = iowa_server_new_incoming_connection(iowaH,
                                                         IOWA_CONN_DATAGRAM,
                                                         newConnection,
                                                         true);

        }
    }

    iowa_close(iowaH);
    close(serverSocket);

    return 0;
}
```

## 4.2 Data types

### 4.2.1 iowa\_supported\_format\_t

```
typedef uint8_t iowa_supported_format_t;

#define IOWA_SUPPORTED_FORMAT_UNKNOWN    0x00
#define IOWA_SUPPORTED_FORMAT_TLV      0x01
#define IOWA_SUPPORTED_FORMAT_JSON     0x02
#define IOWA_SUPPORTED_FORMAT_OLD_TLV  0x04
#define IOWA_SUPPORTED_FORMAT_OLD_JSON  0x08
#define IOWA_SUPPORTED_FORMAT_CBOR     0x10
#define IOWA_SUPPORTED_FORMAT_SENML_JSON 0x20
#define IOWA_SUPPORTED_FORMAT_SENML_CBOR 0x40
```

This contains the possible supported content format. It is an enumeration of the following values:

#### **IOWA\_SUPPORTED\_FORMAT\_UNKNOWN**

Unknown supported format.

#### **IOWA\_SUPPORTED\_FORMAT\_TLV**

TLV supported format.

#### **IOWA\_SUPPORTED\_FORMAT\_JSON**

JSON supported format.

#### **IOWA\_SUPPORTED\_FORMAT\_OLD\_TLV**

Old TLV supported format.

#### **IOWA\_SUPPORTED\_FORMAT\_OLD\_JSON**

Old JSON supported format.

#### **IOWA\_SUPPORTED\_FORMAT\_CBOR**

CBOR format.

#### **IOWA\_SUPPORTED\_FORMAT\_SENML\_JSON**

SenML JSON supported format.

#### **IOWA\_SUPPORTED\_FORMAT\_SENML\_CBOR**

SenML CBOR supported format.

### 4.2.2 iowa\_lwm2m\_protocol\_version\_t

```
typedef enum
{
    IOWA_LWM2M_VERSION_UNDEFINED = 0,
    IOWA_LWM2M_VERSION_1_0,
    IOWA_LWM2M_VERSION_1_1
} iowa_lwm2m_protocol_version_t;
```

This contains the possible LwM2M Enabler version. It is an enumeration of the following values:

#### **IOWA\_LWM2M\_VERSION\_UNDEFINED**

Unknown LwM2M enabler version.

#### **IOWA\_LWM2M\_VERSION\_1\_0**

LwM2M Enabler version 1.0.

#### **IOWA\_LWM2M\_VERSION\_1\_1**

LwM2M Enabler version 1.1.

### 4.2.3 iowa\_client\_t

This structure describes a LwM2M Client known to the LwM2M Server.

```
typedef struct {
    const char          *name;
    uint16_t            id;
    bool                queueMode;
    iowa_supported_format_t supportedFormats;
    const char          *msisdn;
    size_t              objectLinkCount;
    iowa_lwm2m_object_link_t *objectLinkArray;
    uint32_t            lifetime;
    iowa_connection_type_t connectionType;
    bool                secureConnection;
    iowa_lwm2m_protocol_version_t lwm2mVersion;
} iowa_client_t;
```

#### **name**

The unique name of the Client.

#### **id**

The internal ID of the Client. To be used with `iowa_server_dm...()` APIs.

#### **queueMode**

Set to `true` if the LwM2M Client supports the Queue Mode.

#### **supportedFormats**

The content formats supported by the Client.

#### **msisdn**

The MSISDN to which the LwM2M Client is reachable for SMS trigger.

#### **objectLinkCount**

The number of elements in the `objectLinkArray`.

#### **objectLinkArray**

An array containing the Objects and Object Instances registered by the Client.

If an Object has no Instances, `iowa_lwm2m_object_link_t::instanceID` is set to `IOWA_LWM2M_ID_ALL`.

#### **lifetime**

The lifetime of the Client.

#### **connectionType**

The type of the connection on which the Client reach the Server.

#### **secureConnection**

Set to `true` if the connection is encrypted.

#### **lwm2mVersion**

The LwM2M Enabler version used by the Client.

## 4.3 Callbacks

### 4.3.1 iowa\_result\_callback\_t

The device management APIs (`iowa_server_write()`, `iowa_server_dm_exec()`, `iowa_server_dm_create()`, `iowa_server_dm_delete()`, `iowa_server_dm_discover()` and `iowa_server_observe()`) are using an `iowa_result_callback_t` to asynchronously return the result of the operation.

```
typedef void(*iowa_result_callback_t) (iowa_dm_operation_t operation,
                                       uint16_t clientId,
                                       uint16_t objectId,
                                       uint16_t instanceId,
                                       uint16_t resourceId,
                                       iowa_status_t status,
                                       size_t dataCount,
                                       iowa_lwm2m_data_t *dataArray,
                                       void *resultUserData,
                                       iowa_context_t contextP);
```

#### **operation**

The type of command matching this result.

#### **clientId**

The ID of the client targeted by the command.

#### **objectId**

The ID of the Object targeted by the command.

#### **instanceId**

The ID of the Instance targeted by the command. This may be `IOWA_LWM2M_ID_ALL`.

#### **resourceId**

The ID of the Resource targeted by the command. This may be `IOWA_LWM2M_ID_ALL`.

#### **status**

The status of the command or the notification counter if operation is `IOWA_DM_NOTIFY`.

#### **dataCount**

The number of elements in the `dataArray`. This may be 0.

#### **dataArray**

An array containing the Resource values returned by the Client. This may be nil.

#### **resultUserData**

A pointer to application specific data. This is a parameter of the matching `iowa_server_dm_...()` API.

#### **contextP**

The IOWA context on which the device management API (`iowa_server_dm_exec()`) was called.

### 4.3.2 iowa\_monitor\_callback\_t

This is the client state monitoring callback, called when a Lwm2m Client changes its registration to the Server or when a Lwm2m Client connects to the Bootstrap Server.

```
typedef void (*iowa_monitor_callback_t)(const iowa_client_t *clientP,
                                       iowa_state_t state,
                                       void *callbackUserData,
                                       iowa_context_t contextP);
```

#### **clientP**

The information of the Client.



### **state**

The new state of the Client among:

- **IOWA\_STATE\_REGISTERED**: when a new or returning client registers.
- **IOWA\_STATE\_UPDATING**: when a client updates its registration.
- **IOWA\_STATE\_UNREGISTERED**: when a client ends its registration or when the registration expires.
- **IOWA\_STATE\_BOOTSTRAP\_REQUIRED**: when a new or returning client connects to the bootstrap server.
- **IOWA\_STATE\_BOOTSTRAPPING**: when a client is in bootstrapping state.
- **IOWA\_STATE\_BOOTSTRAP\_FAILED**: when the bootstrapping procedure of a client failed.
- **IOWA\_STATE\_BOOTSTRAP\_FINISHED**: when the bootstrapping procedure of a client succeeded.

### **callbackUserData**

A pointer to application specific data. This is a parameter of `iowa_server_configure()`.

### **contextP**

The IOWA context on which `iowa_server_configure()` was called.

When *state* is set to **IOWA\_STATE\_UNREGISTERED**, *clientP* contains only the ID of the former Client.

When *state* is set to **IOWA\_STATE\_BOOTSTRAP\_REQUIRED**, **IOWA\_STATE\_BOOTSTRAPPING**, **IOWA\_STATE\_BOOTSTRAP\_FAILED**, or **IOWA\_STATE\_BOOTSTRAP\_FINISHED**, *clientP* contains only the ID, the name of the Client and the connection type information.

## **4.3.3 iowa\_resource\_type\_callback\_t**

This is the callback called to retrieve the data type of resources of non standard LwM2M Objects.

```
typedef iowa_lwm2m_data_type_t(*iowa_resource_type_callback_t) (uint16_t objectID,
                                                                uint16_t resourceID,
                                                                void *callbackUserData);
```

### **Arguments**

#### **objectID**

The ID of the non standard LwM2M Objects.

#### **resourceID**

The ID of the resource inside the non standard LwM2M Objects.

#### **callbackUserData**

A pointer to application specific data. This is a parameter of `iowa_server_configure()`.

### **Return Value**

The data type of the resource or **IOWA\_LWM2M\_TYPE\_UNDEFINED**.

## **4.3.4 iowa\_verify\_client\_callback\_t**

This is the callback called when LwM2M Clients register to the Server. If the callback returns **IOWA\_COAP\_NO\_ERROR**, the Client is accepted by the Server. Otherwise, to reject a Client the callback has to return **ONLY** the following values:

- **IOWA\_COAP\_400\_BAD\_REQUEST** if the Client is unknown or something does not match.
- **IOWA\_COAP\_409\_CONFLICT** if on LoRaWAN transport, the Client didn't provide its objects list and this list is not present on Server side.

```
typedef iowa_status_t (*iowa_verify_client_callback_t) (const iowa_client_t *clientP,
                                                       iowa_state_t state,
                                                       void *callbackUserData,
                                                       iowa_context_t contextP);
```

#### ***clientP***

The information of the Client.

#### ***state***

The new state of the Client among:

- **IOWA\_STATE\_REGISTERING**: when a new or returning client registers.
- **IOWA\_STATE\_UPDATING**: when a client updates its registration.
- **IOWA\_STATE\_BOOTSTRAP\_REQUIRED**: when a new or returning client connects to the bootstrap server.

#### ***callbackUserData***

A pointer to application specific data. This is a parameter of ['iowa\\_server\\_set\\_verify\\_client\\_callback\(\)'](#).

#### ***contextP***

The IOWA context on which [iowa\\_server\\_configure\(\)](#) was called.

When *state* is set to **IOWA\_STATE\_BOOTSTRAP\_REQUIRED**, *clientP* contains only the ID, the name of the Client and the connection type information.

## 4.4 API

### 4.4.1 iowa\_server\_configure

#### Prototype

```
iowa_status_t iowa_server_configure(iowa_context_t contextP,  
                                   iowa_monitor_callback_t monitorCb,  
                                   iowa_resource_type_callback_t resTypeCb,  
                                   void * callbackUserData);
```

#### Description

`iowa_server_configure()` sets the monitoring callback called when LwM2M Clients register to the Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **monitorCb**

The callback called when Clients update their status. This can be nil.

##### **resTypeCb**

The callback called when parsing received data of non standard LwM2M Objects. This can be nil.

##### **callbackUserData**

A pointer to application specific data. This is passed as argument to `monitorCb` and `resTypeCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`iowa_server.h`

## 4.4.2 iowa\_server\_set\_verify\_client\_callback

### Prototype

```
void iowa_server_set_verify_client_callback(iowa_context_t contextP,
                                           iowa_verify_client_callback_t verifyClientCb,
                                           void *callbackUserData);
```

### Description

`iowa_server_set_verify_client_callback()` sets the verify client callback called when LwM2M Clients register to the Server.

### Arguments

#### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### **verifyClientCb**

The callback called when Clients register. This can be nil.

#### **callbackUserData**

A pointer to application specific data. This is passed as argument to `verifyClientCb`. This can be nil.

### Return Value

None.

### Header File

`iowa_server.h`

### Notes

If the verify client callback is not set, Clients will always be accepted.

### 4.4.3 iowa\_server\_new\_incoming\_connection

#### Prototype

```
iowa_status_t iowa_server_new_incoming_connection(iowa_context_t contextP,  
                                                  iowa_connection_type_t type,  
                                                  void * connP,  
                                                  bool isSecure);
```

#### Description

`iowa_server_new_incoming_connection()` informs the stack of a new incoming connection.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **type**

The type of the new connection. See `iowa_connection_type_t`.

##### **connP**

The new connection of the same user-defined type as the one returned by `iowa_system_connection_open()`.

##### **isSecure**

Set to `true` if the security must be enabled on this connection.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

reading on the new connection failed.

#### Header File

`iowa_server.h`

#### 4.4.4 iowa\_server\_configure\_data\_push

##### Prototype

```
void iowa_server_configure_data_push(iowa_context_t contextP,  
                                   iowa_response_callback_t responseCb,  
                                   void *userDataP);
```

##### Description

`iowa_server_configure_data_push()` enables/disables the Data push operation for all clients.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **responseCb**

The callback called when a client pushes data. If this callback is nil, data push possibility is disabled.

###### **userDataP**

A pointer to application specific data. This is passed as argument to `responseCb`. This can be nil.

##### Return Value

None.

##### Header File

`iowa_server.h`

##### Notes

The `responseCb` will be called with the operation set to **IOWA\_DM\_DATA\_PUSH** and the status code set to **IOWA\_COAP\_205\_CONTENT**.

To be able to use this function, `[LWM2M_DATA_PUSH_SUPPORT][LWM2M_DATA_PUSH_SUPPORT]` must be defined.

#### 4.4.5 iowa\_server\_read

##### Prototype

```
iowa_status_t iowa_server_read(iowa_context_t contextP,
                               uint32_t clientId,
                               size_t uriCount,
                               iowa_lwm2m_uri_t *uriP,
                               iowa_response_callback_t responseCb,
                               void *userDataP)
```

##### Description

`iowa_server_read()` performs a Read operation on Client's URIs.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

###### **uriCount, uriP**

An array of the URIs to read.

###### **responseCb**

The callback called when the reply to this operation is known.

###### **userDataP**

A pointer to application specific data. This is passed as argument to `responseCb`. This can be nil.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_400\_BAD\_REQUEST**

Either:

- `responseCb` is nil.
- `uriCount` is zero or `uriP` is nil.
- `uriP` targets Object ID **IOWA\_LWM2M\_ID\_ALL** and **[LWM2M\_READ\_COMPOSITE\_SUPPORT][LWM2M\_READ\_COMPOSITE\_SUPPORT]** is not defined.
- `uriP` targets Object ID **IOWA\_LWM2M\_ID\_ALL**, **[LWM2M\_READ\_COMPOSITE\_SUPPORT][LWM2M\_READ\_COMPOSITE\_SUPPORT]** is defined, but `uriCount` is not equal to 1.

###### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

###### **IOWA\_COAP\_413\_REQUEST\_ENTITY\_TOO\_LARGE**

The Platform abstraction didn't send all the data. One possible assumption is the packet was too large for the transport.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

A memory allocation failed.

###### **IOWA\_COAP\_501\_NOT\_IMPLEMENTED**

`uriCount` is superior to 1 with **[LWM2M\_READ\_COMPOSITE\_SUPPORT][LWM2M\_READ\_COMPOSITE\_SUPPORT]** not defined.

## IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE

Communication with the LwM2M Client failed.

### Header File

iowa\_server.h

### Notes

Per LwM2M specification, if the Read was successful, the Client will return a **IOWA\_COAP\_205\_CONTENT** status code.

The ability to read several URIs at once is only present in LwM2M version 1.1 or later, this means that to use it **LWM2M\_VERSION\_1\_1\_SUPPORT** must be defined. This feature is only operational on SenML JSON and SenML CBOR data encoding, so **LWM2M\_SUPPORT\_SENML\_JSON** or **LWM2M\_SUPPORT\_SENML\_CBOR** must be defined.

Some LwM2M Clients may not be able to read on several URIs in a single operation. In this case the *resultCb* will be called with an error status, typically **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**.

Per LwM2M specification, when handling a read on several URIs in a single operation, the LwM2M Client treats the request as non-atomic and handles it as best effort. Hence the reply may not contain the values of all the requested URIs.

The *responseCb* will be called with the operation set to **IOWA\_DM\_READ**.



#### 4.4.6 iowa\_server\_observe

##### Prototype

```
iowa_status_t iowa_server_observe(iowa_context_t contextP,
                                   uint32_t clientId,
                                   size_t uriCount,
                                   iowa_lwm2m_uri_t *uriP,
                                   iowa_response_callback_t responseCb,
                                   void *userDataP,
                                   uint16_t *observeIdP);
```

##### Description

`iowa_server_observe()` begins an observation on a Client's URI.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

###### **uriCount, uriP**

An array of the URIs to observe.

###### **responseCb**

The callback called when the reply to this operation is known.

###### **userDataP**

A pointer to application specific data. This is passed as argument to `responseCb`. This can be nil.

###### **observeIdP**

Used to store the ID of the observation.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_400\_BAD\_REQUEST**

Either:

- `responseCb` is nil.
- `observeIdP` is nil.
- `uriCount` is zero or `uriP` is nil.
- `uriP` targets Object ID **IOWA\_LWM2M\_ID\_ALL** and `[LWM2M_READ_COMPOSITE_SUPPORT][LWM2M_READ_COMPOSITE_SUPPORT]` is not defined.
- `uriP` targets Object ID **IOWA\_LWM2M\_ID\_ALL**, `[LWM2M_READ_COMPOSITE_SUPPORT][LWM2M_READ_COMPOSITE_SUPPORT]` is defined, but `uriCount` is not equal to 1.
- At least one `uriP` is invalid: `instanceId` is equal to **IOWA\_LWM2M\_ID\_ALL** but `resourceId` is not equal to **IOWA\_LWM2M\_ID\_ALL**.

###### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

###### **IOWA\_COAP\_412\_PRECONDITION\_FAILED**

Observe was already launched. `observeIdP` is set to the value of the previous observation.

### **IOWA\_COAP\_413\_REQUEST\_ENTITY\_TOO\_LARGE**

The Platform abstraction didn't send all the data. One possible assumption is the packet was too large for the transport.

### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

A memory allocation failed.

### **IOWA\_COAP\_501\_NOT\_IMPLEMENTED**

*uriCount* is superior to 1 with `[LWM2M_OBSERVE_COMPOSITE_SUPPORT][LWM2M_OBSERVE_COMPOSITE_SUPPORT]` not defined.

### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

Communication with the LwM2M Client failed.

## **Header File**

`iowa_server.h`

## **Notes**

Per LwM2M specification, if the Observe was successful, the Client will return a **IOWA\_COAP\_205\_CONTENT** status code with the first notification.

The *responseCb* will be called with the operation set to **IOWA\_DM\_NOTIFY**:

- When the observation is internally deleted, *responseCb* will be called with *status* to **IOWA\_COAP\_202\_DELETED**.
- When the client deregisters or when the connection with the client is lost, *responseCb* will be called with *status* to **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**.
- Some LwM2M Clients may not be able to observe on several URIs in a single operation. In this case the *responseCb* will be called with an error status, typically **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**.

When using an unreliable communication layer, notifications may be lost or arrive out of order.

## 4.4.7 iowa\_server\_observe\_cancel

### Prototype

```
iowa_status_t iowa_server_observe_cancel(iowa_context_t contextP,
                                         uint32_t clientId,
                                         uint16_t observeId);
```

### Description

`iowa_server_observe_cancel()` cancels an observation on a Client.

### Arguments

#### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

#### **observeId**

The ID of the observation as returned by `iowa_server_observe`.

### Return Value

#### **IOWA\_COAP\_NO\_ERROR**

success.

#### **IOWA\_COAP\_404\_NOT\_FOUND**

either:

- *clientId* does not match a known client.
- *observeId* does not match a known observation.

#### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

A memory allocation failed.

#### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

Communication with the LwM2M Client failed.

### Header File

`iowa_server.h`

### Notes

When using an unreliable communication layer, the cancellation request from the LwM2M Server to the LwM2M Client may be lost. However the observation is always cancelled.

#### 4.4.8 iowa\_server\_write

##### Prototype

```
iowa_status_t iowa_server_write(iowa_context_t contextP,
                                uint32_t clientId,
                                size_t dataCount,
                                iowa_lwm2m_data_t *dataArrayP,
                                iowa_response_callback_t responseCb,
                                void *userDataP)
```

##### Description

`iowa_server_write()` performs a Write operation on a Client.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

###### **dataCount, dataArrayP**

The data to write.

###### **responseCb**

The callback called when the reply to this operation is known. This can be nil.

###### **userDataP**

A pointer to application specific data. This is passed as argument to `responseCb`. This can be nil.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_400\_BAD\_REQUEST**

Either:

- `dataCount` is zero or `dataArrayP` is nil.
- `dataArrayP[x].objectID` or `dataArrayP[x].instanceID` or `dataArrayP[x].resourceID` is **IOWA\_LWM2M\_ID\_ALL**.
- `dataArrayP` contains several data with incorrect type.

###### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

###### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- `dataArrayP` contains several data with different `objectID` or `instanceID` but **LWM2M\_SUPPORT\_SENML\_JSON** or **LWM2M\_SUPPORT\_SENML\_CBOR** are not defined.
- `dataArrayP` contains several data with defined timestamp.
- `dataArrayP` contains several data of unsigned integer type which are negative.

###### **IOWA\_COAP\_413\_REQUEST\_ENTITY\_TOO\_LARGE**

The Platform abstraction didn't send all the data. One possible assumption is the packet was too large for the transport.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

### **IOWA\_COAP\_501\_NOT\_IMPLEMENTED**

*dataArrayP* has different *objectID* or *instanceID* with **LWM2M\_WRITE\_COMPOSITE\_SUPPORT**[**LWM2M\_WRITE\_COMPOSITE\_SUPPORT**] not defined.

### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

Communication with the LwM2M Client failed.

### Header File

iowa\_server.h

### Notes

Per LwM2M specification, if the Write was successful, the Client will return a **IOWA\_COAP\_204\_CHANGED** status code.

To be able to write on different *dataArrayP[x].objectID* or *dataArrayP[x].instanceID* at once, **LWM2M\_SUPPORT\_SENML\_JSON** or **LWM2M\_SUPPORT\_SENML\_CBOR** must be defined.

Some LwM2M Clients may not be able to write on different *dataArrayP[x].objectID* or *dataArrayP[x].instanceID* in a single operation. In this case the *responseCb* will be called with an error status, typically **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**.

The *responseCb* will be called with the operation set to **IOWA\_DM\_WRITE**.

#### 4.4.9 iowa\_server\_write\_attributes\_string

##### Prototype

```
iowa_status_t iowa_server_write_attributes_string(iowa_context_t contextP,
                                                uint32_t clientId,
                                                iowa_lwm2m_uri_t *uriP,
                                                const char *attributesStr,
                                                iowa_response_callback_t responseCb,
                                                void *userDataP);
```

##### Description

`iowa_server_write_attributes_string()` performs a Write-Attributes operation on a Client's URI.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

###### **uriP**

The URI targeted by the operation.

###### **attributesStr**

The attributes to write as a query string.

###### **responseCb**

The callback called when the reply to this operation is known. This can be nil.

###### **userDataP**

A pointer to application specific data. This is passed as argument to `responseCb`. This can be nil.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_400\_BAD\_REQUEST** either

`uriP` is nil.

`uriP->objectId` is **IOWA\_LWM2M\_ID\_ALL**.

`uriP->resInstanceId` is not **IOWA\_LWM2M\_ID\_ALL** and **LWM2M\_VERSION\_1\_1\_SUPPORT** is not set.

`attributesStr` is nil or an empty string.

###### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

###### **IOWA\_COAP\_413\_REQUEST\_ENTITY\_TOO\_LARGE**

The Platform abstraction didn't send all the data. One possible assumption is the packet was too large for the transport.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

###### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

Communication with the LwM2M Client failed.

##### Header File

`iowa_server.h`

## Notes

Per LwM2M specification, if the Write-Attributes was successful, the Client will return a **IOWA\_COAP\_204\_CHANGED** status code.

The *responseCb* will be called with the operation set to **IOWA\_DM\_WRITE\_ATTRIBUTES**.

LwM2M defines the following attributes:

Name	Level	Description
pmin	Object, Object Instance, Resource	The minimum period in seconds to wait between notifications.
pmax	Object, Object Instance, Resource	The maximum period in seconds to wait between notifications.
gt	Numerical Resource	An upper threshold. A notification is sent when the resource value crosses this threshold.
lt	Numerical Resource	An lower threshold. A notification is sent when the resource value crosses this threshold.
st	Numerical Resource	A difference minimum in a resource value for a notification to be sent.
epmin	Object, Object Instance, Resource	The minimum sample time in seconds for the observed sensor in LwM2M 1.1 or later.
epmax	Object, Object Instance, Resource	The maximum sample time in seconds for the observed sensor in LwM2M 1.1 or later.

Setting an attribute is in the form `Name "="value` with some constraints:

- `lt value < gt value`
- `lt value + 2 * st value < gt value`
- If `pmax < pmin`, `pmax` is ignored
- `epmax > epmin`

Clearing an attribute is in the form `Name`.

## attributesStr Examples

Receiving a notification every minute at most even if the observed URI did not change: `"pmax=60"`.

Receiving only one notification per hour even if the observed URI changed several times per minute: `"pmin=3600"`.

Receiving exactly one notification every sixty seconds: `"pmin=59&pmax=60"`.

Receiving a notification when the resource value exceeds 95 or falls below 10, and when the resource value returns below 95 or above 10: `"lt=10&gt=95"`.

Clearing the previously set minimum period and setting a maximum period of five minutes: `"pmin&pmax=300"`.

#### 4.4.10 iowa\_server\_dm\_exec

##### Prototype

```
iowa_status_t iowa_server_dm_exec(iowa_context_t contextP,
                                  uint32_t clientID,
                                  uint16_t objectID,
                                  uint16_t instanceID,
                                  uint16_t resourceID,
                                  iowa_result_callback_t resultCb,
                                  void * resultUserData);
```

##### Description

`iowa_server_dm_exec()` performs an Execute operation on a Client's URI.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **clientID**

The ID of the client as reported in the `iowa_monitor_callback_t`.

###### **objectID**

The ID of the Object.

###### **instanceID**

The ID of the instance.

###### **resourceID**

The ID of the resource.

###### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

###### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

###### **IOWA\_COAP\_400\_BAD\_REQUEST**

`objectID`, `instanceID` or `resourceID` is **IOWA\_LWM2M\_ID\_ALL**.

###### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientID` does not match a known client.

##### Header File

`iowa_server.h`

##### Notes

Per LwM2M specification, if the Execute was successful, the Client will return an **IOWA\_COAP\_204\_CHANGED** status code.

Per LwM2M specification, a Server can do an Execute only on an URI in the form `/object/instance/resource`. Thus `instanceID` and `resourceID` cannot be set **IOWA\_LWM2M\_ID\_ALL**.



The *resultCb* will be called with the operation set to **IOWA\_DM\_EXECUTE**.

Evaluation

#### 4.4.11 iowa\_server\_dm\_create

##### Prototype

```
iowa_status_t iowa_server_dm_create(iowa_context_t contextP,
                                    uint32_t clientId,
                                    uint16_t objectId,
                                    uint16_t instanceId,
                                    size_t dataCount,
                                    iowa_lwm2m_data_t *dataArrayP,
                                    iowa_result_callback_t resultCb,
                                    void *resultUserData);
```

##### Description

`iowa_server_dm_create()` performs a Create operation on a Client's URI.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

###### **objectId, instanceId**

The Object Instance targeted by the operation.

###### **dataCount, dataArrayP**

The data to write.

###### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

###### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

###### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- `objectId` or `instanceId` is **IOWA\_LWM2M\_ID\_ALL**.
- `dataCount` is zero or `dataArrayP` is nil.
- `dataArrayP` contains several data with incorrect type.

###### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

###### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

Either:

- `dataArrayP` contains several data with defined timestamp.
- `dataArrayP` contains several data of unsigned integer type which are negative.

## Header File

iowa\_server.h

## Notes

Per LwM2M specification, if the Create was successful, the Client will return a **IOWA\_COAP\_201\_CREATED** status code.

The IDs contained in the data must match *objectId* and *instanceId*.

The *resultCb* will be called with the operation set to **IOWA\_DM\_CREATE**.

Evaluation

#### 4.4.12 iowa\_server\_dm\_delete

##### Prototype

```
iowa_status_t iowa_server_dm_delete(iowa_context_t contextP,
                                     uint32_t clientID,
                                     uint16_t objectID,
                                     uint16_t instanceID,
                                     iowa_result_callback_t resultCb,
                                     void * resultUserData);
```

##### Description

`iowa_server_dm_delete()` performs an Delete operation on a Client's URI.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **clientID**

The ID of the client as reported in the `iowa_monitor_callback_t`.

###### **objectID**

The ID of the Object.

###### **instanceID**

The ID of the instance to delete.

###### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

###### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

###### **IOWA\_COAP\_400\_BAD\_REQUEST**

`objectID` or `instanceID` is **IOWA\_LWM2M\_ID\_ALL**.

###### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientID` does not match a known client.

##### Header File

`iowa_server.h`

##### Notes

Per LwM2M specification, if the Delete was successful, the Client will return an **IOWA\_COAP\_202\_DELETED** status code.

Per LwM2M specification, a Server can do a Delete only on an URI in the form `/object/instance`. Thus `objectID` and `instanceID` cannot be set **IOWA\_LWM2M\_ID\_ALL**.

The `resultCb` will be called with the operation set to **IOWA\_DM\_DELETE**.

### 4.4.13 iowa\_server\_dm\_discover

#### Prototype

```
iowa_status_t iowa_server_dm_discover(iowa_context_t contextP,
                                      uint32_t clientID,
                                      uint16_t objectID,
                                      uint16_t instanceID,
                                      uint16_t resourceID,
                                      iowa_result_callback_t resultCb,
                                      void * resultUserData);
```

#### Description

`iowa_server_dm_discover()` performs a Discover operation on a Client's URI.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **clientID**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### **objectID**

The ID of the Object.

##### **instanceID**

The ID of the instance to delete. This can be `IOWA_LWM2M_ID_ALL`.

##### **resourceID**

The ID of the resource to observe. This can be `IOWA_LWM2M_ID_ALL`.

##### **resultCb**

The callback called when the reply to this operation is known.

##### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- `resultCb` is nil.
- `objectID` is `IOWA_LWM2M_ID_ALL`.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientID` does not match a known client.

#### Header File

`iowa_server.h`

## Notes

Per LwM2M specification, if the Discover was successful, the Client will return an **IOWA\_COAP\_205\_CONTENT** status code.

Per LwM2M specification, a Server can do a Discover on an URI in the forms */object*, */object/instance* or */object/instance/resource*. Thus if *instanceID* is set to **IOWA\_LWM2M\_ID\_ALL**, *resourceID* must be set to **IOWA\_LWM2M\_ID\_ALL**.

The *resultCb* will be called with the operation set to **IOWA\_DM\_DISCOVER**.

Evaluation

#### 4.4.14 iowa\_server\_set\_response\_format

##### Prototype

```
iowa_status_t iowa_server_set_response_format(iowa_context_t contextP,
                                              uint32_t clientID,
                                              iowa_content_format_t multiResourcesFormat,
                                              iowa_content_format_t singleResourceFormat);
```

##### Description

`iowa_server_set_response_format()` sets the content format to use when requesting data from a Client.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **clientID**

The ID of the client as reported in the `iowa_monitor_callback_t`.

###### **multiResourcesFormat**

format to use when requesting several resources.

###### **singleResourceFormat**

format to use when requesting a single resource.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_400\_BAD\_REQUEST**

`multiResourcesFormat` is set to a content format which does not support multiple resources encoding.

###### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientID` does not match a known client.

##### Header File

`iowa_server.h`

##### Notes

By default, IOWA uses LwM2M TLV for all data encodings. If the flag `LWM2M_VERSION_1_0_REMOVE` is used, IOWA uses CBOR for single resource and SenML CBOR for multiples resources.

If the Client does not support the requested content format, it will switch to another one.

#### 4.4.15 iowa\_server\_set\_payload\_format

##### Prototype

```
iowa_status_t iowa_server_set_payload_format(iowa_context_t contextP,
                                             uint32_t clientID,
                                             iowa_content_format_t multiResourcesFormat,
                                             iowa_content_format_t singleResourceFormat);
```

##### Description

`iowa_server_set_payload_format()` sets the content format to use when sending data to a Client.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **clientID**

The ID of the client as reported in the `iowa_monitor_callback_t`.

###### **multiResourcesFormat**

format to use when sending several resources.

###### **singleResourceFormat**

format to use when sending a single resource.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_400\_BAD\_REQUEST**

`multiResourcesFormat` is set to a content format which does not support multiple resources encoding.

###### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientID` does not match a known client.

##### Header File

`iowa_server.h`

##### Notes

By default, IOWA uses LwM2M TLV for all data encodings. If the flag `LWM2M_VERSION_1_0_REMOVE` is used, IOWA uses CBOR for single resource and SenML CBOR for multiples resources.

If the Client does not support the encoding format of the data provided, it will return a **IOWA\_COAP\_415\_UNSUPPORTED\_CONTENT\_FORMAT** error code in the callback of the `iowa_server_write` call.



#### 4.4.16 iowa\_server\_create\_registration\_update\_trigger\_message

##### Prototype

```
size_t iowa_server_create_registration_update_trigger_message(uint16_t serverInstanceId,
                                                             uint8_t **bufferP);
```

##### Description

`iowa_server_create_registration_update_trigger_message()` creates a Registration Update Trigger message.

When receiving a Registration Update Trigger message, a LwM2M Client updates its registration to the targeted LwM2M Server. This mechanism is useful to “wake” up a LwM2M Client which is not longer reachable on the current transport.

##### Arguments

###### ***serverInstanceId***

The Instance ID of the targeted LwM2M Server in the LwM2M Client’s [Server Object][Server Object].

###### ***bufferP***

Used to store the Registration Update Trigger message.

##### Return Value

The length of the buffer in bytes, or 0 in case of an error.

##### Header File

`iowa_server.h`

##### Notes

*bufferP* will be allocated by the `iowa_server_create_registration_update_trigger_message()` function using `iowa_system_malloc()`. It is the caller responsibility to free the buffer.

It is the caller responsibility to send the Registration Update Trigger message to the LwM2M Client, typically using SMS.

#### 4.4.17 iowa\_server\_close\_client\_connection

##### Prototype

```
iowa_status_t iowa_server_close_client_connection(iowa_context_t contextP,  
                                                  uint32_t clientId);
```

##### Description

`iowa_server_close_client_connection()` closes the current connection with a Client.

##### Arguments

###### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

###### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### Return Value

###### **IOWA\_COAP\_NO\_ERROR**

success.

###### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

##### Notes

Depending of the transport used and the connection encryption, the Client will be informed or not of the closing connection.

Some examples:

- UDP / Not secure: Client is **not** informed
- UDP / Secure: Client is informed
- TCP / Not secure: Client is informed
- TCP / Secure: Client is informed

## 5 | Bootstrap Server Mode API Reference

The functions explained below are defined inside the file *include/iowa\_server.h*.

### 5.1 Bootstrap Server pseudo code

```
#include "iowa_server.h"

int main(int argc,
        char *argv[])
{
    iowa_context_t iowaH;
    iowa_status_t result;
    int bootstrapServerSocket;

    /*****
    * Initialization
    */

    bootstrapServerSocket = open_udp_socket();

    iowaH = iowa_init(NULL);

    result = iowa_bootstrap_server_configure(iowaH, client_monitor, NULL);

    /*****
    * "Main loop"
    */

    while (result == IOWA_COAP_NO_ERROR)
    {
        result = iowa_step(iowaH, 10);

        if (isDataAvailable(bootstrapServerSocket))
        {
            void * newConnection;

            newConnection = create_new_connection();
            result = iowa_server_new_incoming_connection(iowaH,
                                                         IOWA_CONN_DATAGRAM,
                                                         newConnection);
        }
    }

    iowa_close(iowaH);
    close(bootstrapServerSocket);

    return 0;
}
```

### 5.2 Callbacks

### 5.2.1 iowa\_bootstrap\_result\_callback\_t

The bootstrap APIs (`iowa_bootstrap_server_write()`, `iowa_bootstrap_server_delete()`, `iowa_bootstrap_server_finish()` and `iowa_bootstrap_server_add_server()`) are using an `iowa_bootstrap_result_callback_t` to asynchronously return the result of the operation.

```
typedef void(*iowa_bootstrap_result_callback_t) (iowa_bootstrap_operation_t operation,
                                                uint16_t clientId,
                                                uint16_t objectId, uint16_t instanceId,
                                                uint16_t resourceId,
                                                iowa_status_t status,
                                                size_t length, uint8_t *buffer,
                                                void *resultUserData,
                                                iowa_context_t contextP);
```

**operation**

The type of command matching this result.

**clientId**

The ID of the client targeted by the command.

**objectId**

The ID of the Object targeted by the command.

**instanceId**

The ID of the Instance targeted by the command. This may be `IOWA_LWM2M_ID_ALL`.

**resourceId**

The ID of the Resource targeted by the command. This may be `IOWA_LWM2M_ID_ALL`.

**status**

The status of the command.

**length**

The length of the payload when the *operation* is `IOWA_BOOTSTRAP_DISCOVER`.

**buffer**

The payload containing the CoRE Link information when the *operation* is `IOWA_BOOTSTRAP_DISCOVER`.

**resultUserData**

A pointer to application specific data. This is a parameter of the matching `iowa_bootstrap_server...()` API.

**contextP**

The IOWA context on which the bootstrap API was called.

## 5.3 API

### 5.3.1 iowa\_bootstrap\_server\_configure

#### Prototype

```
iowa_status_t iowa_bootstrap_server_configure(iowa_context_t contextP,  
                                              iowa_monitor_callback_t monitorCb,  
                                              void *callbackUserData);
```

#### Description

`iowa_bootstrap_server_configure()` sets the monitoring callback called when LwM2M Clients connect to the LwM2M Bootstrap Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **monitorCb**

The callback called when Clients update their status. This can be nil.

##### **callbackUserData**

A pointer to application specific data. This is passed as argument to `monitorCb` and `resTypeCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

#### Header File

`iowa_server.h`

### 5.3.2 iowa\_bootstrap\_server\_new\_incoming\_connection

#### Prototype

```
iowa_status_t iowa_bootstrap_server_new_incoming_connection(iowa_context_t contextP,
                                                           iowa_connection_type_t type,
                                                           void * connP,
                                                           bool isSecure);
```

#### Description

`iowa_bootstrap_server_new_incoming_connection()` informs the stack of a new incoming connection.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **type**

The type of the new connection. See `iowa_connection_type_t`.

##### **connP**

The new connection of the same user-defined type as the one returned by `iowa_system_connection_open()`.

##### **isSecure**

Set to `true` if the security must be enabled on this connection.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

reading on the new connection failed.

#### Header File

`iowa_server.h`

### 5.3.3 iowa\_bootstrap\_server\_read

#### Prototype

```
iowa_status_t iowa_bootstrap_server_read(iowa_context_t contextP,
                                         uint32_t clientId,
                                         iowa_lwm2m_uri_t *uriP,
                                         iowa_response_callback_t responseCb,
                                         void *userDataP);
```

#### Description

`iowa_bootstrap_server_read()` performs a Bootstrap Read operation on a Client's URI.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### **uriP**

The URI targeted by the operation.

##### **responseCb**

The callback called when the reply to this operation is known.

##### **userDataP**

A pointer to application specific data. This is passed as argument to `responseCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- `objectId` or `instanceId` is **IOWA\_LWM2M\_ID\_ALL**.
- `uriP` is nil.
- `responseCb` is nil.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- `uriP` cannot target Root, Resource or Resource Instance level
- `uriP->objectId` must target the [Server Object][Server Object] or the [Access Control List Object][Access Control List Object].

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`iowa_server.h`

## Notes

Per LwM2M specification, if the Bootstrap Read was successful, the Client will return a **IOWA\_COAP\_205\_CONTENT** status code.

The *responseCb* will be called with the operation set to **IOWA\_BOOTSTRAP\_READ**.

Evaluation



### 5.3.4 iowa\_bootstrap\_server\_write

#### Prototype

```
iowa_status_t iowa_bootstrap_server_write(iowa_context_t contextP,
                                         uint32_t clientId,
                                         uint16_t objectId,
                                         uint16_t instanceId,
                                         size_t dataCount,
                                         iowa_lwm2m_data_t *dataArray,
                                         iowa_bootstrap_result_callback_t resultCb,
                                         void *resultUserData);
```

#### Description

`iowa_bootstrap_server_write()` performs a Bootstrap Write operation on a Client's URI.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### **objectId**

The Object targeted by the operation.

##### **instanceId**

The Instance object targeted by the operation.

##### **dataCount**

The number of data to write.

##### **dataArray**

The data to write.

##### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

##### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

either:

- `objectId` or `instanceId` is **IOWA\_LWM2M\_ID\_ALL**.
- `dataCount` is zero or `dataArray` is nil.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

Either:

- *dataArrayP* contains several data with defined timestamp.
- *dataArrayP* contains several data with unsigned integer which are negative.

### Header File

iowa\_server.h

### Notes

Per LwM2M specification, if the Bootstrap Write was successful, the Client will return a **IOWA\_COAP\_204\_CHANGED** status code.

The IDs contained in the data must match *objectId* and *instanceId*.

The *resultCb* will be called with the operation set to **IOWA\_BOOTSTRAP\_WRITE**.

Evaluation

### 5.3.5 iowa\_bootstrap\_server\_delete

#### Prototype

```
iowa_status_t iowa_bootstrap_server_delete(iowa_context_t contextP,
                                           uint32_t clientId,
                                           uint16_t objectId,
                                           uint16_t instanceId,
                                           iowa_bootstrap_result_callback_t resultCb,
                                           void *resultUserData);
```

#### Description

`iowa_bootstrap_server_delete()` performs a Bootstrap Delete operation on a Client's URI.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### **objectId**

The ID of the Object to delete. This can be **IOWA\_LWM2M\_ID\_ALL**.

##### **instanceId**

The ID of the instance to delete. This can be **IOWA\_LWM2M\_ID\_ALL**.

##### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

##### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

#### Header File

`iowa_server.h`

#### Notes

Per LwM2M specification, if the Bootstrap Delete was successful, the Client will return an **IOWA\_COAP\_204\_CHANGED** status code.

Per LwM2M specification, the Bootstrap Server can request the Client to delete all Objects and Object Instances (except for the Bootstrap Server account) in a single operation by setting `objectId` and `instanceId` to **IOWA\_LWM2M\_ID\_ALL**.

The `resultCb` will be called with the operation set to **IOWA\_BOOTSTRAP\_DELETE**.

### 5.3.6 iowa\_bootstrap\_server\_discover

#### Prototype

```
iowa_status_t iowa_bootstrap_server_discover(iowa_context_t contextP,
                                             uint32_t clientId,
                                             uint16_t objectId,
                                             iowa_bootstrap_result_callback_t resultCb,
                                             void * resultUserData);
```

#### Description

`iowa_bootstrap_server_discover()` performs a Bootstrap Discover operation on a Client's URI.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### **objectId**

The ID of the Object targeted by the operation. This can be **IOWA\_LWM2M\_ID\_ALL**.

##### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

##### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

#### Header File

`iowa_server.h`

#### Notes

Per LwM2M specification, the Bootstrap Discover operation only returns the list of Objects and Object Instances with some attributes: LwM2M Enabler version ("lwm2m="), Short Server ID ("ssid="), and LwM2M Server URI ("uri=").

The `resultCb` will be called with the operation set to **IOWA\_BOOTSTRAP\_DISCOVER**.

### 5.3.7 iowa\_bootstrap\_server\_finish

#### Prototype

```
iowa_status_t iowa_bootstrap_server_finish(iowa_context_t contextP,
                                           uint32_t clientId,
                                           iowa_bootstrap_result_callback_t resultCb,
                                           void *resultUserData);
```

#### Description

`iowa_bootstrap_server_finish()` performs a Bootstrap Server operation on a Client's URI.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

##### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

#### Header File

`iowa_server.h`

#### Notes

Per LwM2M specification, if the Bootstrap Finish was successful, the Client will return an **IOWA\_COAP\_204\_CHANGED** status code. Otherwise the Client will return an **IOWA\_COAP\_406\_NOT\_ACCEPTABLE** status code.

The `resultCb` will be called with the operation set to **IOWA\_BOOTSTRAP\_FINISH**.

### 5.3.8 iowa\_bootstrap\_server\_add\_server

#### Prototype

```
iowa_status_t iowa_bootstrap_server_add_server(iowa_context_t contextP,
                                              uint32_t clientId,
                                              uint16_t shortServerId,
                                              const char *uri,
                                              uint32_t lifetime,
                                              iowa_security_data_t *securityDataP,
                                              iowa_bootstrap_result_callback_t resultCb,
                                              void *resultUserData);
```

#### Description

`iowa_bootstrap_server_add_server()` adds the proper Security and Server object to the client to configure a Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### **shortServerId**

The short ID of the Server.

##### **uri**

The URI of the Server.

##### **lifetime**

The lifetime in seconds of the registration.

##### **securityDataP**

The security data to use to connect properly to the Server.

##### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

##### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

`uri` is nil or `shortServerId` is 0 or **IOWA\_LWM2M\_ID\_ALL**.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

#### Header File

`iowa_server.h`

## Notes

`iowa_bootstrap_server_add_server()` calls internally `iowa_bootstrap_server_discover` to retrieve the [Security Object][Security Object] Instances and [Server Object][Server Object] Instances, then calls `iowa_bootstrap_server_write` two times to write a new [Security Object][Security Object] Instance and a new [Server Object][Server Object] Instance.

The `resultCb` will be called with the operation set to **IOWA\_BOOTSTRAP\_ADD\_SERVER**.

Per LwM2M specification, if adding the server was successful, the Client will return an **IOWA\_COAP\_204\_CHANGED** status code.

Evaluation

### 5.3.9 iowa\_bootstrap\_server\_remove\_server

#### Prototype

```
iowa_status_t iowa_bootstrap_server_remove_server(iowa_context_t contextP,
                                                  uint32_t clientId,
                                                  uint16_t shortServerId,
                                                  iowa_bootstrap_result_callback_t
                                                  resultCb,
                                                  void *resultUserData);
```

#### Description

`iowa_bootstrap_server_remove_server()` removes the proper Security and [Server Object][Server Object] Instance associated to the Short Server ID from a client.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### **shortServerId**

The short ID of the Server.

##### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

##### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

`shortServerId` is 0 or **IOWA\_LWM2M\_ID\_ALL**.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

#### Header File

`iowa_server.h`

#### Notes

`iowa_bootstrap_server_remove_server()` calls internally `iowa_bootstrap_server_discover` to retrieve the [Security Object][Security Object] Instances and [Server Object][Server Object] Instances, then calls `iowa_bootstrap_server_delete` two times to delete a [Security Object][Security Object] Instance and a [Server Object][Server Object] Instance.

The `resultCb` will be called with the operation set to **IOWA\_BOOTSTRAP\_REMOVE\_SERVER**.

Per LwM2M specification:



- if removing the server was successful, the Client will return an **IOWA\_COAP\_202\_DELETED** status code.

If the [[Security Object](#)][Security Object] Instance and/or the [[Server Object](#)][Server Object] Instance associated to the LwM2M Server have not been found after the Discover operation, the result callback will be called with the following parameters:

- objectId: **IOWA\_LWM2M\_SECURITY\_OBJECT\_ID** or **IOWA\_LWM2M\_SERVER\_OBJECT\_ID**
- instancelid: **IOWA\_LWM2M\_ID\_ALL**
- resourcelid: **IOWA\_LWM2M\_ID\_ALL**
- status: **IOWA\_COAP\_404\_NOT\_FOUND**

Evaluation

### 5.3.10 iowa\_bootstrap\_server\_add\_bootstrap\_server

#### Prototype

```
iowa_status_t iowa_bootstrap_server_add_bootstrap_server(
    iowa_context_t contextP,
    uint32_t clientId,
    const char *uri,
    int32_t clientHoldOff,
    uint32_t bootstrapAccountTimeout,
    iowa_security_data_t *securityDataP,
    iowa_bootstrap_result_callback_t resultCb,
    void *resultUserData
);
```

#### Description

`iowa_bootstrap_server_add_bootstrap_server()` adds the proper Security object to the client to configure a Bootstrap Server.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### **uri**

The URI of the Server.

##### **clientHoldOff**

The number of seconds to wait before initiating a Client Initiated Bootstrap.

##### **bootstrapAccountTimeout**

Time to wait by the client before to purge the LwM2M Bootstrap-Server Account.

##### **securityDataP**

The security data to use to connect properly to the Server.

##### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

##### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

`uri` is nil.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

##### **IOWA\_COAP\_413\_REQUEST\_ENTITY\_TOO\_LARGE**

The Platform abstraction didn't send all the data. One possible assumption is the packet was too large for the transport.

## Header File

iowa\_server.h

## Notes

`iowa_bootstrap_server_add_bootstrap_server()` calls internally `iowa_bootstrap_server_discover` to retrieve the [Security Object][Security Object] Instances, then `iowa_bootstrap_server_write` to write a [Security Object][Security Object] Instance.

The *resultCb* will be called with the operation set to **IOWA\_BOOTSTRAP\_ADD\_BOOTSTRAP\_SERVER**.

Per LwM2M specification, if adding the bootstrap server was successful, the Client will return an **IOWA\_COAP\_204\_CHANGED** status code.

Evaluation

### 5.3.11 iowa\_bootstrap\_server\_remove\_bootstrap\_server

#### Prototype

```
iowa_status_t iowa_bootstrap_server_remove_bootstrap_server(
    iowa_context_t contextP,
    uint32_t clientId,
    iowa_bootstrap_result_callback_t resultCb,
    void *resultUserData
);
```

#### Description

`iowa_bootstrap_server_remove_bootstrap_server()` removes the proper [Security Object][Security Object] Instance associated to the Bootstrap Server from a client.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **clientId**

The ID of the client as reported in the `iowa_monitor_callback_t`.

##### **resultCb**

The callback called when the reply to this operation is known. This can be nil.

##### **resultUserData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

##### **IOWA\_COAP\_404\_NOT\_FOUND**

`clientId` does not match a known client.

#### Header File

`iowa_server.h`

#### Notes

`iowa_bootstrap_server_remove_bootstrap_server()` calls internally `iowa_bootstrap_server_discover` to retrieve the [Security Object][Security Object] Instances, then calls `iowa_bootstrap_server_delete` to delete a [Security Object][Security Object] Instance.

The `resultCb` will be called with the operation set to **IOWA\_BOOTSTRAP\_REMOVE\_BOOTSTRAP\_SERVER**.

Per LwM2M specification:

- if the removing the bootstrap server was successful, the Client will return an **IOWA\_COAP\_202\_DELETED** status code.

If the [Security Object][Security Object] Instance associated to the LwM2M Bootstrap Server have not been found after the Discover operation, the result callback will be called with the following parameters:

- `objectId`: **IOWA\_LWM2M\_SECURITY\_OBJECT\_ID**

- instanceId: **IOWA\_LWM2M\_ID\_ALL**
- resourceId: **IOWA\_LWM2M\_ID\_ALL**
- status: **IOWA\_COAP\_404\_NOT\_FOUND**

Evaluation

## 6 | CoAP API Reference

The functions explained below are defined inside the file *include/iowa\_coap.h*.

### 6.1 CoAP client pseudo code

```
#include "iowa_coap.h"

int main(int argc,
        char *argv[])
{
    iowa_context_t iowaH;
    iowa_coap_peer_t *peerP;
    iowa_status_t result;

    /*****
    * Initialization
    */

    iowaH = iowa_init(NULL);

    peerP = iowa_coap_peer_new(iowaH,
                              "coap://coap.example.com",
                              IOWA_SEC_NONE,
                              NULL,
                              prv_coapEventCallback, NULL);

    result = iowa_coap_peer_connect(iowaH, peerP);

    /*****
    * "Main loop"
    */

    while (result == IOWA_COAP_NO_ERROR)
    {
        result = iowa_step(iowaH, 10);
    }

    iowa_coap_peer_delete(iowaH, peerP);

    iowa_close(iowaH);
    close(serverSocket);

    return 0;
}

void prv_coapEventCallback(iowa_coap_peer_t *fromPeer,
                          iowa_coap_event_t event,
                          void *userData,
                          iowa_context_t contextP)
{
    if (event == COAP_EVENT_CONNECTED)
    {

```

```

        iowa_status_t result;

        // Sending a GET on "/test"

        result = iowa_coap_peer_get(contextP, fromPeer, "/test", prv_resultCallback,
            userData);
    }
    else if (event == COAP_EVENT_DISCONNECTED)
    {
        iowa_coap_peer_delete(contextP, fromPeer);
    }
}

void prv_resultCallback(iowa_coap_peer_t *fromPeer,
    uint8_t code,
    iowa_coap_message_t *responseP,
    void *userData,
    iowa_context_t contextP)
{
    printf("Result for GET: %u.%02u.\r\n", (code & 0xFF) >> 5, (code & 0x1F));

    if (code == IOWA_COAP_205_CONTENT)
    {
        printf("Payload: %.*s", responseP->payloadLength, responseP->payload);
    }
}

```

## 6.2 Data types

### 6.2.1 iowa\_coap\_peer\_t

```
typedef struct _iowa_coap_peer_t iowa_coap_peer_t;
```

`iowa_coap_peer_t` is an opaque type describing a CoAP peer.

### 6.2.2 iowa\_coap\_peer\_event\_t

```
typedef enum
{
    COAP_EVENT_UNDEFINED = 0,
    COAP_EVENT_CONNECTED,
    COAP_EVENT_DISCONNECTED
} iowa_coap_peer_event_t;
```

`iowa_coap_peer_event_t` contains the possible events that can be reported by a CoAP peer.

### 6.2.3 iowa\_coap\_message\_t

```
typedef struct _iowa_coap_message_t iowa_coap_message_t;
```

`iowa_coap_message_t` is an opaque type describing a CoAP message.

## 6.2.4 iowa\_coap\_setting\_id\_t

```
typedef uint8_t iowa_coap_setting_id_t;
```

### 6.2.4.1 Possible Values

#### IOWA\_COAP\_SETTING\_ACK\_TIMEOUT

The RFC7252 ACK\_TIMEOUT value as an *uint8\_t*.

#### IOWA\_COAP\_SETTING\_MAX\_RETRANSMIT

The RFC7252 MAX\_RETRANSMIT value as an *uint8\_t*.

#### IOWA\_COAP\_SETTING\_URI\_LENGTH

The length in bytes of the URI as a *size\_t*. This is a read-only setting.

#### IOWA\_COAP\_SETTING\_URI

The URI as a *char \**. This is a read-only setting. The passed argument must point to a buffer of at least the size indicated by

**IOWA\_COAP\_SETTING\_URI\_LENGTH**.



## 6.3 Callbacks

### 6.3.1 iowa\_coap\_result\_callback\_t

The CoAP APIs `iowa_coap_peer_get()` and `iowa_coap_block_request_next()` are using an `iowa_coap_result_callback_t` to asynchronously return the result of the operation.

```
typedef void(*iowa_coap_result_callback_t)(iowa_coap_peer_t *fromPeer,
                                           uint8_t code,
                                           iowa_coap_message_t *messageP,
                                           void *userData,
                                           iowa_context_t contextP);
```

**fromPeer**

The CoAP peer we sent the request to.

**code**

The Code of the CoAP Message or the result of the transmission.

**messageP**

The received CoAP Message. This can be nil.

**userData**

The `iowa_coap_peer_get()` or `iowa_coap_block_request_next()` parameter.

**contextP**

The IOWA context on which the CoAP API (`iowa_coap_peer_get()` or `iowa_coap_block_request_next()`) was called.

### 6.3.2 iowa\_coap\_peer\_event\_callback\_t

```
typedef void(*iowa_coap_peer_event_callback_t)(iowa_coap_peer_t *fromPeer,
                                                iowa_coap_peer_event_t event,
                                                void *userData,
                                                iowa_context_t contextP);
```

**fromPeer**

The CoAP peer which generated the event.

**event**

The event generated by the peer. See `iowa_coap_peer_event_t`.

**userData**

The `iowa_coap_peer_new()` parameter.

**contextP**

The IOWA context on which `iowa_coap_peer_new()` was called.

## 6.4 API

### 6.4.1 iowa\_coap\_peer\_new

#### Prototype

```
iowa_coap_peer_t *iowa_coap_peer_new(iowa_context_t contextP,
                                     const char *uri,
                                     iowa_security_mode_t securityMode,
                                     iowa_coap_peer_event_callback_t eventCb,
                                     void *callbackUserData);
```

#### Description

`iowa_coap_peer_new()` creates a new CoAP peer.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **uri**

The URI of the CoAP peer.

##### **securityMode**

The security to use with this CoAP peer. See `[iowa_security_mode_t][iowa_security_mode_t]`.

##### **eventCb**

The callback to call when this CoAP peer generates an event.

##### **callbackUserData**

A pointer to application specific data. This is passed as argument to `messageCb` and `eventCb`. This can be nil.

#### Return Value

A pointer to an `iowa_coap_peer_t` in case of success or NULL in case of memory allocation error, invalid URI, or if `iowa_system_connection_open()` returned an error.

#### Header File

`iowa_coap.h`

## 6.4.2 iowa\_coap\_peer\_delete

### Prototype

```
void iowa_coap_peer_delete(iowa_context_t contextP,  
                           iowa_coap_peer_t *peerP);
```

### Description

`iowa_coap_peer_delete()` closes a CoAP peer.

### Arguments

#### ***contextP***

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### ***peerP***

The CoAP peer to close. This can be nil.

### Header File

`iowa_coap.h`

### 6.4.3 iowa\_coap\_peer\_configuration\_set

#### Prototype

```
iowa_status_t iowa_coap_peer_configuration_set(iowa_context_t contextP,
                                              iowa_coap_peer_t *peerP,
                                              iowa_coap_setting_id_t settingId,
                                              void *argP);
```

#### Description

`iowa_coap_peer_configuration_set()` configures the settings of a CoAP peer.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **peerP**

The CoAP peer to configure.

##### **settingId**

The setting to set. See `iowa_coap_setting_id_t`.

##### **argP**

A pointer to the setting value. Dependent on `settingId`.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`peerP` is nil.

##### **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**

`settingId` is not valid for this peer type.

##### **IOWA\_COAP\_422\_UNPROCESSABLE\_ENTITY**

`peerP` is of an unsupported type.

#### Header File

`iowa_coap.h`

## 6.4.4 iowa\_coap\_peer\_configuration\_get

### Prototype

```
iowa_status_t iowa_coap_peer_configuration_get(iowa_context_t contextP,
                                              iowa_coap_peer_t *peerP,
                                              iowa_coap_setting_id_t settingId,
                                              void *argP);
```

### Description

`iowa_coap_peer_configuration_get()` retrieves the value of a setting of a CoAP peer.

### Arguments

#### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### **peerP**

The CoAP peer to retrieve the setting from.

#### **settingId**

The setting to retrieve. See `iowa_coap_setting_id_t`.

#### **argP**

A pointer to store the setting value. Dependent on `settingId`.

### Return Value

#### **IOWA\_COAP\_NO\_ERROR**

success.

#### **IOWA\_COAP\_402\_BAD\_OPTION**

`peerP` is nil.

#### **IOWA\_COAP\_405\_METHOD\_NOT\_ALLOWED**

`settingId` is not valid for this peer type.

#### **IOWA\_COAP\_422\_UNPROCESSABLE\_ENTITY**

`peerP` is of an unsupported type.

### Header File

`iowa_coap.h`

### 6.4.5 iowa\_coap\_peer\_connect

#### Prototype

```
iowa_status_t iowa_coap_peer_connect(iowa_context_t contextP,  
                                     iowa_coap_peer_t *peerP);
```

#### Description

`iowa_coap_peer_connect()` opens a connection with a CoAP peer.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **peerP**

The CoAP peer to connect.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`peerP` is nil.

#### Header File

`iowa_coap.h`

#### Notes

The actual result of the connection is indicated by a call to the `iowa_coap_peer_event_callback_t` associated to the CoAP peer.

## 6.4.6 iowa\_coap\_peer\_disconnect

### Prototype

```
void iowa_coap_peer_disconnect(iowa_context_t contextP,  
                              iowa_coap_peer_t *peerP);
```

### Description

`iowa_coap_peer_disconnect()` closes a connection with a CoAP peer.

### Arguments

#### ***contextP***

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### ***peerP***

The CoAP peer to disconnect. This can be nil.

### Return Value

None.

### Header File

`iowa_coap.h`

### Notes

The actual result of the disconnection is indicated by a call to the `iowa_coap_peer_event_callback_t` associated to the CoAP peer.

## 6.4.7 iowa\_coap\_peer\_get

### Prototype

```
iowa_status_t iowa_coap_peer_get(iowa_context_t contextP,
                                iowa_coap_peer_t *peerP,
                                const char *path,
                                const char *query,
                                iowa_coap_result_callback_t resultCb,
                                void *userData);
```

### Description

`iowa_coap_peer_get()` sends a CoAP GET request to a CoAP peer.

### Arguments

#### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### **peerP**

The CoAP peer to send the request to.

#### **path**

The path component of the uri to retrieve. This can be nil.

#### **query**

The query component of the uri to retrieve. This can be nil.

#### **resultCb**

The callback to call when a reply is received or when the transmission fails.

#### **userData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

### Return Value

#### **IOWA\_COAP\_NO\_ERROR**

success.

#### **IOWA\_COAP\_402\_BAD\_OPTION**

`peerP` or `resultCb` is nil.

#### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- `iowa_system_gettime()` returned an error.

#### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

either:

- the CoAP peer is not connected.
- `iowa_system_connection_send()` returned an error.

### Header File

`iowa_coap.h`



## 6.4.8 iowa\_coap\_message\_get\_payload

### Prototype

```
size_t iowa_coap_message_get_payload(iowa_coap_message_t *messageP,  
                                     iowa_content_format_t *formatP,  
                                     uint8_t **payloadP);
```

### Description

`iowa_coap_message_get_payload()` retrieves the pointer to the payload of a CoAP message.

### Arguments

#### ***messageP***

the CoAP message to inspect.

#### ***formatP***

OUT. The content format of the payload. This can be nil.

#### ***payloadP***

OUT. A pointer to the payload of the message. This can be nil.

### Return Value

The length in bytes of the payload.

### Notes

If the CoAP message has no Content-format option, *formatP* will be set to **IOWA\_CONTENT\_FORMAT\_UNSET**.

### Header File

`iowa_coap.h`

## 6.4.9 iowa\_coap\_message\_get\_block\_info

### Prototype

```
iowa_status_t iowa_coap_message_get_block_info(iowa_coap_message_t *messageP,
                                              uint32_t *numberP,
                                              bool *moreP,
                                              uint16_t *sizeP);
```

### Description

`iowa_coap_message_get_block_info()` retrieves block information in a CoAP message.

### Arguments

#### ***messageP***

the CoAP message to inspect.

#### ***numberP***

OUT. the block number.

#### ***moreP***

OUT. true if there are more blocks coming.

#### ***sizeP***

OUT. the size of the block.

### Return Value

#### **IOWA\_COAP\_NO\_ERROR**

success.

#### **IOWA\_COAP\_404\_NOT\_FOUND**

if the CoAP message has no block information.

### Header File

`iowa_coap.h`

## 6.4.10 iowa\_coap\_block\_request\_next

### Prototype

```
iowa_status_t iowa_coap_block_request_next(iowa_context_t contextP,
                                           iowa_coap_peer_t *peerP,
                                           iowa_coap_message_t *messageP,
                                           iowa_coap_message_callback_t resultCb,
                                           void *userData);
```

### Description

When receiving a reply with a block option, `iowa_coap_block_request_next()` requests the next block from the CoAP peer.

### Arguments

#### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

#### **peerP**

The CoAP peer to send the message to.

#### **messageP**

The CoAP message containing the previous block.

#### **resultCb**

The callback to call when the next block is received, or if an error occurs.

#### **userData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

### Return Value

#### **IOWA\_COAP\_NO\_ERROR**

success.

#### **IOWA\_COAP\_402\_BAD\_OPTION**

`peerP` or `messageP` is nil.

#### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- `iowa_system_gettime()` returned an error.

#### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

either:

- the CoAP peer is not connected.
- `iowa_system_connection_send()` returned an error.

### Notes

IOWA must be compiled with the flag `[IOWA_COAP_BLOCK_SUPPORT][Additional flags]`.

### Header File

`iowa_coap.h`

### 6.4.11 iowa\_coap\_block\_request\_block\_number

#### Prototype

```
uint8_t iowa_coap_block_request_block_number(iowa_context_t contextP,
                                             iowa_coap_peer_t *peerP,
                                             iowa_coap_message_t *messageP,
                                             uint32_t blockNumber,
                                             iowa_coap_result_callback_t resultCb,
                                             void *userData);
```

#### Description

When receiving a reply with a block option, `iowa_coap_block_request_block_number()` requests a specific block from the CoAP peer.

#### Arguments

##### **contextP**

An `iowa_context_t` as returned by `iowa_init()`. Not checked at runtime.

##### **peerP**

The CoAP peer to send the message to.

##### **messageP**

The CoAP message containing the block transfer.

##### **blockNumber**

the block number.

##### **resultCb**

The callback to call when the block is received, or if an error occurs.

##### **userData**

A pointer to application specific data. This is passed as argument to `resultCb`. This can be nil.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_402\_BAD\_OPTION**

`peerP` or `messageP` is nil.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

either:

- a memory allocation failed.
- `iowa_system_gettime()` returned an error.

##### **IOWA\_COAP\_503\_SERVICE\_UNAVAILABLE**

either:

- the CoAP peer is not connected.
- `iowa_system_connection_send()` returned an error.

#### Notes

IOWA must be compiled with the flag `[IOWA_COAP_BLOCK_SUPPORT]`[Additional flags].

## Header File

iowa\_coap.h

Evaluation

## 6.5 Helper Functions

### 6.5.1 iowa\_coap\_uri\_parse

#### Prototype

```
iowa_status_t iowa_coap_uri_parse(const char *uri,
                                   iowa_connection_type_t *typeP,
                                   char **hostnameP,
                                   char **portP,
                                   char **pathP,
                                   char **queryP,
                                   bool *isSecureP);
```

#### Description

`iowa_coap_uri_parse()` parses an URI and may return its type, hostname, port, path, and query.

#### Arguments

##### ***uri***

the URI to parse.

##### ***typeP***

OUT. the connection type.

##### ***hostnameP***

OUT. the hostname. This can be nil.

##### ***portP***

the port. This can be nil.

##### ***pathP***

OUT. the path. This can be nil.

##### ***queryP***

OUT. the path. This can be nil.

##### ***isSecureP***

OUT. inform if the connection is secured.

#### Return Value

##### **IOWA\_COAP\_NO\_ERROR**

success.

##### **IOWA\_COAP\_400\_BAD\_REQUEST**

*uri* is nil.

##### **IOWA\_COAP\_406\_NOT\_ACCEPTABLE**

either:

- the URI schema has not been recognized.
- the URI format is invalid.

##### **IOWA\_COAP\_500\_INTERNAL\_SERVER\_ERROR**

a memory allocation failed.

#### Header File

`iowa_coap.h`

## 7 | Utils API Reference

The functions explained below are defined inside the file *include/iowa\_utils.h*.

### 7.1 Data types

#### 7.1.1 iowa\_list\_t

List structure used by the List APIs.

```
typedef struct _iowa_list_t
{
    struct _iowa_list_t *nextP;
} iowa_list_t;
```

##### nextP

Pointer to the next element in the list.

## 7.2 Callbacks

### 7.2.1 iowa\_list\_node\_free\_callback\_t

This is the list node free callback, called to free a node.

```
typedef void(*iowa_list_node_free_callback_t) (void *nodeP);
```

#### ***nodeP***

The node to free.



### 7.2.2 iowa\_list\_node\_find\_callback\_t

This is the list node find callback, called to find a node.

```
typedef bool(*iowa_list_node_find_callback_t) (void *nodeP,  
                                              void *criteria);
```

***nodeP***

The current node in the list.

***criteria***

The criteria to match.

Evaluation

## 7.3 API

### 7.3.1 iowa\_utils\_base64\_get\_encoded\_size

#### Prototype

```
size_t iowa_utils_base64_get_encoded_size(size_t rawBufferLen);
```

#### Description

`iowa_utils_base64_get_encoded_size()` calculates the length of a Base64 buffer based on a raw buffer represented by its length.

#### Arguments

##### ***rawBufferLen***

The length of the raw buffer.

#### Return Value

The length of the Base64 buffer.

#### Header File

`iowa_utils.h`

### 7.3.2 iowa\_utils\_base64\_get\_decoded\_size

#### Prototype

```
size_t iowa_utils_base64_get_decoded_size(uint8_t *base64Buffer,  
                                           size_t base64BufferLen);
```

#### Description

`iowa_utils_base64_get_decoded_size()` calculates the length of a raw buffer based on a Base64 buffer.

#### Arguments

***base64Buffer***

The Base64 buffer.

***base64BufferLen***

The length of the Base64 buffer.

#### Return Value

The length of the raw buffer. If any error the length will be 0.

#### Header File

`iowa_utils.h`

### 7.3.3 iowa\_utils\_base64\_encode

#### Prototype

```
size_t iowa_utils_base64_encode(uint8_t * rawBuffer,
                                size_t rawBufferLen,
                                uint8_t * base64Buffer,
                                size_t base64BufferLen);
```

#### Description

`iowa_utils_base64_encode()` encodes a raw buffer using Base64.

#### Arguments

##### ***rawBuffer***

The raw buffer.

##### ***rawBufferLen***

The length of the raw buffer.

##### ***base64Buffer***

The preallocated Base64 buffer.

##### ***base64BufferLen***

The length of the preallocated Base64 buffer.

#### Return Value

The length of the encoded buffer. If any error the length will be 0.

#### Header File

`iowa_utils.h`

### 7.3.4 iowa\_utils\_base64\_decode

#### Prototype

```
size_t iowa_utils_base64_decode(uint8_t * base64Buffer,
                                size_t base64BufferLen,
                                uint8_t * rawBuffer,
                                size_t rawBufferLen);
```

#### Description

`iowa_utils_base64_decode()` decodes a Base64 buffer into a raw buffer.

#### Arguments

##### ***base64Buffer***

The Base64 buffer.

##### ***base64BufferLen***

The length of the Base64 buffer.

##### ***rawBuffer***

The preallocated raw buffer.

##### ***rawBufferLen***

The length of the preallocated raw buffer.

#### Return Value

The length of the decoded buffer. If any error the length will be 0.

#### Header File

`iowa_utils.h`

### 7.3.5 iowa\_utils\_uri\_to\_sensor

#### Prototype

```
iowa_sensor_t iowa_utils_uri_to_sensor(iowa_lwm2m_uri_t *uriP);
```

#### Description

`iowa_utils_uri_to_sensor()` converts an `iowa_lwm2m_uri_t` into an `iowa_sensor_t`.

#### Arguments

##### *uriP*

Uri to convert.

#### Return Value

The corresponding `iowa_sensor_t` or `IOWA_INVALID_SENSOR_ID` in case of error.

#### Header File

`iowa_utils.h`

### 7.3.6 iowa\_utils\_sensor\_to\_uri

#### Prototype

```
iowa_lwm2m_uri_t iowa_utils_sensor_to_uri(iowa_sensor_t id);
```

#### Description

`iowa_utils_sensor_to_uri()` converts an `iowa_sensor_t` into an `iowa_lwm2m_uri_t`.

#### Arguments

***id***

Id to convert.

#### Return Value

An `iowa_lwm2m_uri_t`.

#### Header File

`iowa_utils.h`

### 7.3.7 iowa\_utils\_list\_add

#### Prototype

```
iowa_list_t * iowa_utils_list_add(iowa_list_t *headP,  
                                  iowa_list_t *nodeP);
```

#### Description

`iowa_utils_list_add()` adds a node to a list.

#### Arguments

***headP***

Head of the current list.

***nodeP***

Node to add to the list.

#### Return Value

The list with the new element.

#### Header File

`iowa_utils.h`



### 7.3.8 iowa\_utils\_list\_remove

#### Prototype

```
iowa_list_t * iowa_utils_list_remove(iowa_list_t *headP,  
                                     iowa_list_t *nodeP);
```

#### Description

`iowa_utils_list_remove()` removes a node from a list.

#### Arguments

***headP***

Head of the current list.

***nodeP***

Node to remove from the list.

#### Return Value

The updated list.

#### Header File

`iowa_utils.h`

### 7.3.9 iowa\_utils\_list\_free

#### Prototype

```
void iowa_utils_list_free(iowa_list_t *headP,  
                          iowa_list_node_free_callback_t freeCb);
```

#### Description

`iowa_utils_list_free()` adds a node to a list.

#### Arguments

##### ***headP***

List to free.

##### ***freeCb***

Callback used to free the list.

#### Return Value

None.

#### Header File

`iowa_utils.h`

### 7.3.10 iowa\_utils\_list\_find

#### Prototype

```
iowa_list_t * iowa_utils_list_find(iowa_list_t *headP,  
                                   iowa_list_node_find_callback_t findCb,  
                                   void *criteriaP);
```

#### Description

`iowa_utils_list_find()` finds a node in a list.

#### Arguments

##### ***headP***

List to search on.

##### ***findCb***

Callback used to find the node in the list.

##### ***criteriaP***

Criteria used to find the node in the list.

#### Return Value

The node if found else NULL.

#### Header File

`iowa_utils.h`

### 7.3.11 iowa\_utils\_list\_find\_and\_remove

#### Prototype

```
iowa_list_t * iowa_utils_list_find_and_remove(iowa_list_t *headP,
                                              iowa_list_node_find_callback_t findCb,
                                              void *criteriaP,
                                              iowa_list_t **nodeP);
```

#### Description

`iowa_utils_list_find_and_remove()` finds a node in a list and removes it.

#### Arguments

##### **headP**

List to search on.

##### **findCb**

Callback used to find the node in the list.

##### **criteriaP**

Criteria used to find the node in the list.

##### **nodeP**

OUT. Node removed from the list. Can be nil.

#### Return Value

The updated list.

#### Header File

`iowa_utils.h`

## 7.4 Example: Linked List usage

When declaring the linked list data structure, the first member **must** be a pointer. This pointer will contain the address of the next element of the list.

Example:

```
struct myData
{
    struct myData *nextP;    // Used by the linked list functions
    char          *aString;
    int           anInt;
};
```

The head of the list is a pointer to your data structure.

Example:

```
struct myData *listHead;
```

You can now add or remove elements to the list by using the functions `iowa_utils_list_add()` and `iowa_utils_list_remove()`.

But to avoid compiler warnings or multiple cast making the code unreadable, the following macros can be used:

- IOWA\_UTILS\_LIST\_ADD(H, N)
- IOWA\_UTILS\_LIST\_REMOVE(H, N)
- IOWA\_UTILS\_LIST\_FREE(H, F)
- IOWA\_UTILS\_LIST\_FIND(H, F, C)
- IOWA\_UTILS\_LIST\_FIND\_AND\_REMOVE(H, F, C, N)

Instead of calling:

```
listHead = (struct myData *)iowa_utils_list_add((iowa_list_t *)listHead, (iowa_list_t *)
newDataP);
```

You can use:

```
listHead = (struct myData *)IOWA_UTILS_LIST_ADD(listHead, newDataP);
```

## 8 | IOWA Components

This section describes how you can replace some parts of IOWA by your own.

Unlike the APIs, the internals functions described in this section may change between IOWA releases.

### 8.1 Overview

Internally, IOWA is organized in several components as shown in the figure below.

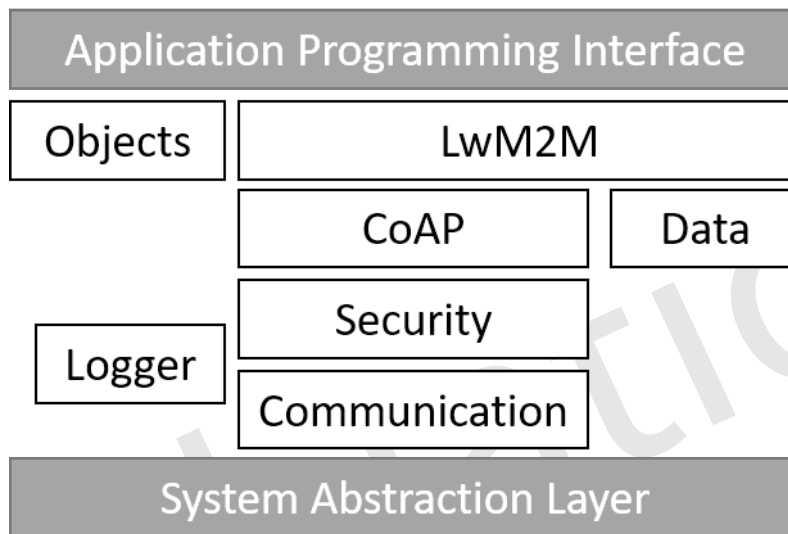


Figure 8.1: Components

#### Objects

Found in *src/objects*. This is an aggregate of the implementations of the LwM2M Objects supported by IOWA.

#### LwM2M

Found in *src/lwm2m*. The LightweightM2M engine of IOWA.

#### Data

Found in *src/data*. This component handles the serialization and deserialization of the various data formats used by IOWA.

#### CoAP

Found in *src/coap*. The Constrained Application Protocol stack of IOWA.

#### Security

Found in *src/security*. In charge of the security sessions management and a wrapper to the [Security layers][Security layers].

#### Communication

Found in *src/comm*. A wrapper to the `iowa_system_connection_...()` functions to aggregate all the connections opened by the other components.

#### Logger

Found in *src/logger*. Formats then outputs the IOWA logs through `iowa_system_trace()`.

## 8.2 Logger Component

The functions explained below are defined inside the file *include/iowa\_logger.h*.

### 8.2.1 Presentation

To use your logger layer, you have to set the define `[IOWA_LOGGER_USER][IOWA_LOGGER_USER]` and implement the following functions.

```
void iowa_log(uint8_t part,
             uint8_t level,
             const char *functionName,
             unsigned int line,
             const char *message);

void iowa_log_arg(uint8_t part,
                 uint8_t level,
                 const char *functionName,
                 unsigned int line,
                 const char *message, ...);

void iowa_log_buffer(uint8_t part,
                    uint8_t level,
                    const char *functionName,
                    unsigned int line,
                    const char *message,
                    const uint8_t *buffer,
                    size_t bufferLength);

void iowa_log_arg_buffer(uint8_t part,
                        uint8_t level,
                        const char *functionName,
                        unsigned int line,
                        const char *message,
                        const uint8_t *buffer,
                        size_t bufferLength,
                        ...);
```

## 8.2.2 Functions

### 8.2.2.1 iowa\_log

**Prototype**

```
void iowa_log(uint8_t part,  
             uint8_t level,  
             const char *functionName,  
             unsigned int line,  
             const char *message);
```

**Description** `iowa_log()` writes a log message to the output.

**Arguments**

***part***

Log part.

***level***

Log level.

***functionName***

Name of the function from where the Log has been called.

***line***

Line from where the Log has been called.

***message***

String to display.

**Return Value** None.

**Header File** `iowa_logger.h`



### 8.2.2.2 iowa\_log\_arg

#### Prototype

```
void iowa_log_arg(uint8_t part,  
                 uint8_t level,  
                 const char *functionName,  
                 unsigned int line,  
                 const char *message, ...);
```

**Description** `iowa_log_arg()` writes a log message to the output with specifier arguments.

#### Arguments

***part***

Log part.

***level***

Log level.

***functionName***

Name of the function from where the Log has been called.

***line***

Line from where the Log has been called.

***message***

String to display.

...

Format specifiers which are replaced by the values specified in additional arguments.

**Return Value** None.

**Header File** `iowa_logger.h`

### 8.2.2.3 iowa\_log\_buffer

#### Prototype

```
void iowa_log_buffer(uint8_t part,
                    uint8_t level,
                    const char *functionName,
                    unsigned int line,
                    const char *message,
                    const uint8_t *buffer,
                    size_t bufferLength);
```

**Description** `iowa_log_buffer()` writes a buffer with a log message to the output.

#### Arguments

##### ***part***

Log part.

##### ***level***

Log level.

##### ***functionName***

Name of the function from where the Log has been called.

##### ***line***

Line from where the Log has been called.

##### ***message***

String to display.

##### ***buffer***

Buffer.

##### ***bufferLength***

Buffer size.

**Return Value** None.

**Header File** `iowa_logger.h`

#### 8.2.2.4 iowa\_log\_arg\_buffer

##### Prototype

```
void iowa_log_arg_buffer(uint8_t part,
                        uint8_t level,
                        const char *functionName,
                        unsigned int line,
                        const char *message,
                        const uint8_t *buffer,
                        size_t bufferLength,
                        ...);
```

**Description** `iowa_log_arg_buffer()` writes a buffer with a log message to the output with specifier arguments.

##### Arguments

###### **part**

Log part.

###### **level**

Log level.

###### **functionName**

Name of the function from where the Log has been called.

###### **line**

Line from where the Log has been called.

###### **message**

String to display.

###### **buffer**

Buffer.

###### **bufferLength**

Buffer size.

...

Format specifiers which are replaced by the values specified in additional arguments.

**Return Value** None.

**Header File** `iowa_logger.h`

## 9 | Deprecated API Reference

Deprecated APIs are no longer supported which means that regression bugs will not be fixed. These APIs should not be used anymore. You are strongly advised to use their replacement.

Note that deprecated API can be removed in a future IOWA release.

### 9.1 Deprecated Compilation Flags

#### 9.1.1 IOWA\_SINGLE\_CONNECTION\_MODE

Support only one connection at a time. Useful for constrained devices in a single LwM2M Server environment.

##### 9.1.1.1 Notes

This define has no more effect when set.

#### 9.1.2 LWM2M\_SINGLE\_SERVER\_MODE

This is only relevant when IOWA is in Client mode. When set, the Client supports only one Server configuration at a time. Useful for constrained devices in a single LwM2M Server environment.

This define cannot be set when `[LWM2M_BOOTSTRAP][LWM2M_BOOTSTRAP]` is already set.

##### 9.1.2.1 Notes

This define has no more effect when set.

#### 9.1.3 LWM2M\_OLD\_CONTENT\_FORMAT\_SUPPORT

During the development of the Lightweight M2M protocol, some LwM2M products were released. These products were using temporary numbers for the content format of the LwM2M payload. Setting this flag allows IOWA to interact with these old implementations. It should be seldom required.

##### 9.1.3.1 Notes

This define has no more effect when set. By default, the old content formats code are always supported.

#### 9.1.4 IOWA\_LORAWAN\_MINIMAL\_SUPPORT

Minimal support for LoRaWAN transport. URI scheme is in the form "lorawan://". The Endpoint will not be able to send its Objects list in the Registration message and Registration Update message if needs. This define should only be considered if the code size of IOWA is a concern.

##### 9.1.4.1 Notes

This define has no more effect when set.

#### 9.1.5 LWM2M\_NOTIFICATION\_QUEUE\_SIZE

This is only relevant when IOWA is in Client mode. When set, this define sets the maximum stored notification values when the notification are not able to reach the Server. The values are saved in RAM by IOWA internally inside a First In First Out Queue. The maximum values stored is per observation.

By default, when the define is not set, the value is 4.

### 9.1.5.1 Notes

This define has no more effect when set.

## 9.1.6 LWM2M\_STORAGE\_QUEUE\_PEEK\_SUPPORT

When a LwM2M Server observing some resources is not reachable, the LwM2M Client stores the notifications until the connectivity is restored. By default, IOWA stores the last notifications in memory. When this flag is set, IOWA discharges the storage of these notifications to the platform. New version using a peek/remove mechanism instead of a dequeue mechanism.

This feature requires the system abstraction functions `iowa_system_queue_create()`, `iowa_system_queue_enqueue()`, `iowa_system_queue_peek()`, `iowa_system_queue_remove()`, and `iowa_system_queue_delete()` to be implemented.

### 9.1.6.1 Notes

This define will be later replaced by **LWM2M\_STORAGE\_QUEUE\_SUPPORT**. It means the Storage Queue with Peek behavior will be the default.

## 9.1.7 LwM2M features removal

LwM2M mandatory features can be removed depending of the use case. Removing a feature should only be done to reduce the code size of IOWA on constrained devices, and should not be considered for other case:

### **LWM2M\_READ\_OPERATION\_REMOVE**

Remove the ability to handle a Read Operation. Only relevant for LwM2M Client mode. When this flag is set, **LWM2M\_OBSERVE\_OPERATION\_REMOVE** and **LWM2M\_WRITE\_ATTRIBUTES\_OPERATION\_REMOVE** are also set.

### **LWM2M\_DISCOVER\_OPERATION\_REMOVE**

Remove the ability to handle a Discover Operation. Only relevant for LwM2M Client mode.

### **LWM2M\_WRITE\_OPERATION\_REMOVE**

Remove the ability to handle a Write Operation. Only relevant for LwM2M Client mode.

### **LWM2M\_WRITE\_ATTRIBUTES\_OPERATION\_REMOVE**

Remove the ability to handle a Write-Attributes Operation. Only relevant for LwM2M Client mode.

### **LWM2M\_EXECUTE\_OPERATION\_REMOVE**

Remove the ability to handle a Execute Operation. Only relevant for LwM2M Client mode.

### **LWM2M\_CREATE\_OPERATION\_REMOVE**

Remove the ability to handle a Create Operation. Only relevant for LwM2M Client mode.

### **LWM2M\_DELETE\_OPERATION\_REMOVE**

Remove the ability to handle a Delete Operation. Only relevant for LwM2M Client mode.

### **LWM2M\_OBSERVE\_OPERATION\_REMOVE**

Remove the ability to handle a Observe Operation. Only relevant for LwM2M Client mode. When this flag is set, **LWM2M\_WRITE\_ATTRIBUTES\_OPERATION\_REMOVE** is also set.

### **LWM2M\_OPAQUE\_CONTENT\_FORMAT\_REMOVE**

Remove the ability to decode/encode the Opaque Content Format.

### **LWM2M\_TEXT\_CONTENT\_FORMAT\_REMOVE**

Remove the ability to decode/encode the Plain Text Content Format.

### 9.1.7.1 Notes

These defines have no more effect when set.

# A | Appendix A

IOWA reuses code from various open source projects.

- wakaama is copyrighted by Intel Corporation and others and reused under the Eclipse Distribution License 1.0.

Copyright (c) 2013, 2014 Intel Corporation

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*Eclipse Distribution License - v 1.0*

Copyright (c) 2007, Eclipse Foundation, Inc. and its licensors.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Eclipse Foundation, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,

BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

- mbed TLS is copyrighted by ARM Limited under the Apache License, Version 2.0.
- tinydtls is copyrighted by Olaf Bergmann and others and reused under the Eclipse Distribution License 1.0.

Copyright (c) 2011, 2012, 2013, 2014, 2015, 2016 Olaf Bergmann (TZI) and others. All rights reserved. This program and the accompanying materials are made available under the terms of the Eclipse Public License v1.0 and Eclipse Distribution License v. 1.0 which accompanies this distribution. The Eclipse Public License is available at <http://www.eclipse.org/legal/epl-v10.html> and the Eclipse Distribution License is available at <http://www.eclipse.org/org/documents/edl-v10.php>.

Contributors: \* Olaf Bergmann - initial API and implementation \* Hauke Mehrtens - memory optimization, ECC integration

- OMA LwM2M objects are copyrighted by Open Mobile Alliance.

Copyright 2017 Open Mobile Alliance All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The above license is used as a license under copyright only. Please reference the OMA IPR Policy for patent licensing terms: <http://www.openmobilealliance.org/ipr.html>

- AT Command LwM2M object is copyrighted by Cisco.

### BSD 3-Clause License

Copyright (c) 2017, Cisco All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.