SmartHub.ai

SmartHub INFER

IoT & Edge Management

API Guide

Find more about our products and solutions at https://www.smarthub.ai/

Table of contents

1 Introduction to the API

The SmartHub INFER IoT Center API offers a way for third-party systems to interact with SmartHub INFER IoT Center.

SmartHub INFER IoT Center provides programmable REST APIs to integrate with your existing enterprise solution. With the SmartHub INFER IoT Center APIs, you can create, view, edit, and delete various SmartHub INFER IoT Center entities such as Devices, Campaigns, Alerts, Notifications, Groups, and Users, programmatically. All the core features of the SmartHub INFER IoT Center provide REST APIs.

1.1 Terminology

This document uses the following terms.

API: API is an acronym for Application Programming Interface. It is a name used to refer to a special framework some web applications or services provide which allows a user to connect to the system and perform some number of discrete actions such as running functions, requesting data, or updating information.

Call: A call is another name for a request or a communication sent by a user to the API, in the form of a URL string, which invokes a specific action on one particular endpoint, and can also include additional parameters or values.

REST: REST is an acronym for Representational State Transfer. It is a form of software architecture that is primarily used for designing a web service.

HTTP: HTTP is an acronym for Hyper-Text Transport Protocol. It is one of the key architectural components behind how web-based content on the Internet is accessed through web browsers.

JSON: JSON is an acronym for JavaScript Object Notation. It is a format for information, based on the JavaScript language, that is intended for consumption by a programmed function.

Method: HTTP provides support for four methods which each describe a type of result a user might want to achieve through a given communication with a web server or API. The four methods are:

GET	GET is used for retrieving or querying records from a system.
PUT	PUT is used for inserting or creating records into a system.
POST	POST is used for updating existing records in a system.
DELETE	DELETE is used for removing records from a system.

Note: The required permissions that are listed for each API are for invoking the API operation. Some APIs require additional permissions to complete the operation.

1.2 APIs provided by SmartHub INFER

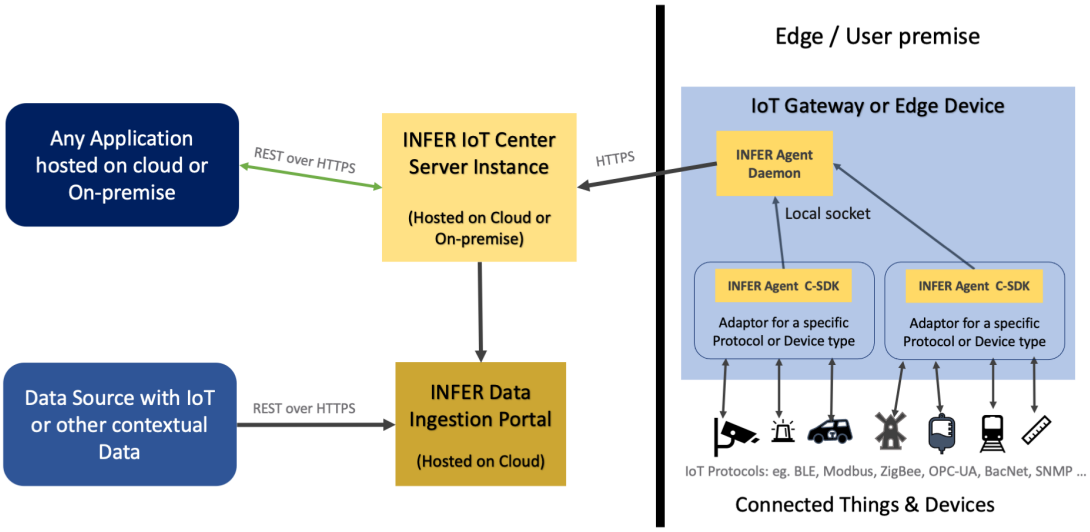


Figure 1-1. SmartHub INFER API Overview Diagram

SmartHub INFER provides 3 APIs for 3 different types of Integration as depicted in the above diagram.

1.2.1 Server APIs

This is a set of REST APIs that are provided by SmartHub INFER Server to enable applications to read data stored in INFER, write data into INFER and control actions performed by INFER on IoT & Edge devices. All of the functionality provided by SmartHub INFER Console (UI) is implemented using this same set of REST APIs. Note that the consumer of these APIs needs to have inbound HTTPS access to the INFER instance - regardless of whether it deployed on-premise or in the cloud.

1.2.2 Data Ingestion APIs

This is a set of REST APIs provided by **INFER Data Ingestion Portal**, which is hosted in the cloud with an unchanging URL. It that enables third party applications to send device-related data, Alert data or other contextual data into a central place which then gets forwarded to its intended INFER Instance. The INFER instance can be running in the Cloud or on-premise.

1.2.3 Agent APIs

This is a set of C Functions that is provided as an SDK library to enable Edge Adapters or other applications running at the Edge to inject IoT data into INFER via INFER Agent Daemon. The SDK library can be consumed by C/C++, Python, Go or other language programs that implement a particular protocol to interface with devices.

1.3 User Authentication

Use the following APIs to create and issue an authentication token for a user.

1.3.1 Acquire Token

Header

If the user name is available across multiple organizations, enter the following header:

```
x-org-domain-name:<domain>
```

API `/api/tokens`

Method `GET`

Required Parameters `None`

Response

```
{
    "accessToken": “string”,
    "expiresInSecs": "1543317540",
    "accessTokenExpiresAt": "1543317540",
    "refreshToken": “string”,
    "refreshTokenExpiresAt": "1544519940"
}
```

1.3.2 Issue Access Token Using Refresh Token

API `/api/tokens/refresh`

Method `GET`

Required Parameters `None`

Response

```
{
  "accessToken": "string",
  "accessTokenExpiresAt": "1543317540"
  "expiresInSecs": 0,
  "refreshToken": null
}
```

1.4 Organizations

Use the following api to list all organizations in your scope.

1.4.1 List Organizations

API ‘/api/organizations’

Method `GET`

Required Parameters `None`

Response

```
{
  "pageInfo": {
    "totalPages": "string",
    "totalElements": "string",
    "page": 0,
    "pageSize": 0
  },
```

```
  "tenants": [
    {
      "id": "string",
      "name": "string",
      "parentId": "string",
      "status": "ACTIVE",
      "ancestors": [
        "string"
      ],
      "orgId": "string",
      "lastUpdatedBy": "string",
      "createdBy": "string",
      "lastUpdatedTime": "string",
      "createdTime": "string",
      "updateVersion": 0
    }
  ]
}
```


1.5 Device Authentication

Use the following APIs to issue device token and device credentials.

1.5.1 Create Device Credential

Required Permissions

You must have the **Create Device Credential** permission to perform this operation.

API `api/device-credentials/{id}`

Method `POST`

Input Examples

JWT_NATIVE	`{}`
PROPERTY_NATIVE	`{"requestParams":"{\"DeviceKey\":\"1234\"}"}`
TPM_NATIVE	`{"requestParams":"{\"tpm_ek\":\"123456\"}"}`

Response


```
{
  "credentials": "string"
}
```


1.5.2 Get Device Token

Required Permissions

You must have the **Get Device Token** permission to perform this operation.

API `/api/device-tokens`

Header `x-device-auth`

Enter the device credential that you created in the Create Device Credential API.

Method `GET`

Required Parameters

Name	Type	Description
`id`	string	Device ID

Response

```
{
  "deviceId": "string",
  "accessToken": "string"
}
```

1.6 HTTP Response Codes

The following table lists the response codes used by SmartHub INFER IoT Center. The response codes adhere to the standard HTTP and REST conventions.

HTTP Status Codes

Status Code	Error Message
200	Success
401	Unauthorized
403	Permission denied
404	Not found

1.7 API Headers

APIs must include the following headers:

1.7.1 API Version

```
Content-Type: application/json
Accept: application/json;api-version=<api-version>
```

To get the current API version, use the following API:

API `/api/versions`

Method `GET`

Sample Response

```
{
  "currentApiVersion": "0.2",
  "supportedApiVersions": [
    "0.1",
    "0.2"
  ]
}
```

1.7.2 User Access Token

You require a user access token to perform API operations.

```
Authorization : {UserAccessToken}
```

For information about generating user access tokens, see the User Authentication section.

1.7.3 Set the Current Organization ID

Use this header to set the current organization ID for which you want to run the APIs.

```
x-current-org-id:<orgId>
```

1.8 Restricted Characters

The following characters are restricted when creating a template name, device name, custom property, and metric name.

1.8.1 Template Name

```
< > % $ ( ) { }
```

1.8.2 Device Name

```
< > % $ ( ) { } [ ]
```

1.8.3 Custom Property

```
< > . % $ ( ) { }
```

1.8.4 Metric Name

```
: { } & "
```

2 Server APIs

This is a set of REST APIs that are provided by SmartHub INFER Server to enable other applications to read data stored in INFER, write data into INFER and control actions performed by INFER on IoT & Edge devices. Note that the consumer of these APIs needs to have inbound HTTPS access to the INFER instance - regardless of whether it deployed on-premise or in the cloud.

2.1 Swagger UI

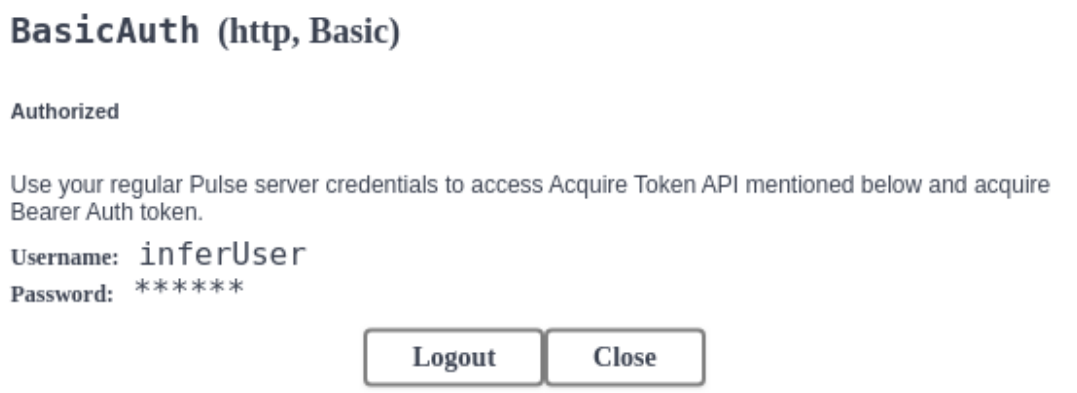
SmartHub INFER's Server APIs are OpenAPI compliant and provides a Swagger UI as part the INFER server instance. A Swagger UI provides detailed documentation for all of the RESTful APIs offered by the Server. It also provides the ability to invoke the APIs interactively with a live server connection. More details about Swagger UI can be found at https://swagger.io

Users of SmartHub INFER can access the swagger UI by pointing their browser to `https://<INFER-SERVER-FQDN>/openapi/index.html` where `<INFER-SERVER-FQDN>` is the Fully Qualified Domain Name or IP address of the INFER Server Instance.

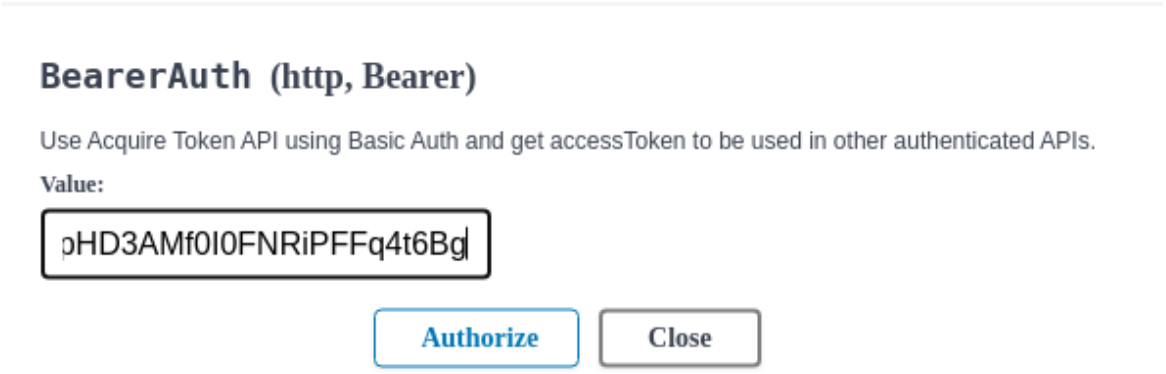
2.2 Using the Server API

To invoke APIs on the Server, one needs to be authenticated first. This is done by invoking the following APIs in sequence.

1. Ensure that the **Servers** drop-down list at the top-left of the page is showing the correct FQDN for the INFER instance you would like to connect to.
2. Click the **Authorize** button in the top right corner to perform `Basic Auth` to the INFER server. Click **Close** once you are logged in.



3. Next, get the API versions supported by the server by invoking API Version call. For all subsequent calls, ensure that a supported version is provided in the **Accept header**. In the Swagger UI, the api-version value is automatically added to the header but when invoking the APIs otherwise, one needs to send this parameter in the header as shown in the curl example.
4. Acquire an **Access Bearer token**, that you can use for all subsequent calls in the session. This can be done by invoking the Acquire Token call. The response will provide two tokens - `accessToken` and `refreshToken` .
5. Authorize all other REST calls using the value of `accessToken` as BearerAuth token. You can do this by clicking the gray-colored lock at the right, for any of the REST APIs to open the popup and enter the value of `accessToken` returned by Acquire Token call.



6. Additionally and **optionally**, you can set the scope of the Apis for a particular (sub-)organization. This can be done by providing `orgId` as a header in the CurrentOrgId-Header by opening the popup as mentioned above. If there's no value provided, the scope of organization is automatically identified by the server from the BearerAuth token. OrgId can be found in the organization list which could be retrieved from the List Organizations API.

2.3 Types of Server APIs

The rest of this section provides a high-level overview of the Server APIs along with some sample use cases.

- **Device Management** - APIs for performing operations on devices, device templates, device authentication, device commands, and files
- **Campaign Management** - Create, Modify, Delete, Start, Stop and other operations on Campaigns. Create, Delete and management of Packages
- **Alerting** - APIs to get, create, update, and delete alerts and alert definitions
- **Identity and Access Management (IAM)** - APIs to perform tenant management, sub-org management, role management, user management, and group management operations
  - **Organization Management** - Create, view, update, and delete an organization
  - **User Management** - APIs to create, fetch, update, and delete users
  - **Role Management** - APIs to create, fetch, update, and delete roles
  - **Group Management** - APIs to create, fetch, update, and delete user groups
  - **Permission Management** - API to fetch permissions
  - **Organization Settings** - APIs to view and update your organization settings

    - **Password Management** - APIs to generate a password recovery link and reset the password
    - **Token Management** - APIs to generate access and refresh tokens
- **Notification** - APIs for Notification Destinations, Notification Definitions, and Notification Instances
    - **Notification Definition** - APIs to create, update, get, and delete notification definitions
    - **Notification Instances** - APIs to retrieve notification instances
    - **System Notifications** - APIs to view system notifications. System notifications are generated when there is a system downtime. The notifications are sent to the users through email or displayed on the UI
- **Certificate Management** - APIs to create, update and delete certificates
- **Advanced Search** - APIs to create, get, update, and delete a filtered device list
- **Metric APIs** - APIs to query metrics
- **Audit APIs** - APIs to get audit logs, get audit types, and get entity types

3 Data Ingestion APIs

This is a set of REST APIs provided by **SmartHub INFER Data Ingestion Portal**, that enables third party applications to send device, Alert or other contextual data into any SmartHub INFER. The Data Ingestion portal is hosted in the cloud with an unchanging URL and can scale to accomodate data ingestion for any number of INFER instances.

As the Data Ingestion Portal is a single public entry point for data to be sent into any INFER instance, users of this portal need to authenticate themselves using an **API Key** and tag the data using a combination of **Client Instance Identifier** and **Device ID**. More details are provided below.

3.1 API Specification

URL Endpoint

`https://vmi.us02.smarthubai.net`

API

`/ingest`

Header

This API uses an **API key** as its authentication mechanism and it needs to be specified in the request header for every invocation. Please contact SmartHub.ai to get your API key.

```
x-api-key:  <YOUR KEY>
```

Request Method

`POST`

Request JSON structure

```
{
  "viid":"<CLIENT INSTANCE IDENTIFIER>",
  "diid":"<DEVICE ID>",
  "timeStamp":"",
  “payLoads”: [
   {
      "payLoadType" : "Properties",
      "deviceType" : "Camera",
      "location" : "",
      "cameraName" : "",
      "cameraIP" : "",
      "modelNumber" : "",
      "serialNumber" : "",
      "firmware" : "",
      "alertStatus" : "",
      "severity" : "",
      "securityProfile" : "",
      "alertStatus" : "",
      "severity" : ""
    },
    {
      "payLoadType" : "Metrics",
      "CPU-Usage" : 23.4,
      "Memory-Usage" : 87.34,
      "Disk-Usage" : 15,
      "wlan0-up" : true,
      "eth0-status" : "Offline"
```

```
    },
    {
      "payLoadType" : "Event",
      "deviceType" : "Locker",
      "Locker ID" : "lkr_45255407047802",
      "Event-Type" : "Locker Opened",
      "Event-Timestamp" : "2021-01-11T15:00:19.003Z"
    },
    {
      "payLoadType" : "Alert",
      "alertType" : "tailgating",
      “alertID” : “214495-bafc-24-4fcdaa“,
      "location" : "",
      "cameraName" : "",
      "cameraIP" : "",
      "modelNumber" : "",
      "serialNumber" : "",
      "firmware" : "",
      "alertStatus" : "",
      "severity" : "",
      "securityProfile" : "",
      "alertStatus" : "",
      "severity" : ""
   }
  ]
}
```

payLoads is an array of one or more payloads

payLoadType can have one of the following values:

Properties	When properties of the device needs to sent at a specific frequency
Metrics	When metrics collected from the device needs to sent in a time-series mode
Event	Specific events related to the application/device, only if applicable
Alert	Alerts in the source application to be sent as alerts into SmartHub INFER platform

These are the **Mandatory keys** that need to be present in the JSON for every call:

viid	Identifies the client instance for which data is sent
diid	Identifies the Device or Asset to the SmartHub INFER platform
timeStamp	Timestamp of the occurrence in the source system
payLoads	Array of payload objects
payLoadType	Identifies the payload type (One for each of the payLoad object in the array)

3.2 API Invocation example using curl

```
curl --request POST \
  --url https://vmi.us02.smarthubai.net/ingest \
  --header 'content-type: application/json;api-version=0.17' \
  --header 'x-api-key:  <YOUR KEY> ' \
  --data '<JSON INPUT>'
```

4 Agent APIs

This is a C-SDK (Software Development Kit) that is provided by SmartHub INFER Agent to enable other various device-specific Adapters to interact with the INFER Agent running at the Edge.

Header

```
#include <iotcAgent.h>
```

4.1 Data Structures

4.1.1 IotcApplicationId

`IotcApplicationId` represents the application identifier.

Application identifier is any string with a maximum length of `IOTC_APP_ID_SIZE - 1` . It is used to identify an application uniquely during an exchange of data between the edge application and the server side application. Use the reverse domain name notation, such as `ai.smarthub.iotc.agent`

Data Fields

`char id[IOTC_APP_ID_SIZE]`

- Holds the actual characters of the identifiers.

4.1.2 IotcAllowedMetricInfo

Stores the allowed metric information.

`char metricName[IOTC_METRIC_NAME_SIZE]`

- Name of the metric.

`IotcMetricType metricType`

- Type of metric.

4.1.3 IotcAllowedMetricSet

Set of allowed metrics of a device.

Data Fields

`size_t size`

- Size of the allowed metric set.

`IotcAllowedMetricInfo *allowedMetricInfo`

- Set of allowed metrics.

4.1.4 IotcBooleanValue

IotcBooleanValue represents the boolean type metric data point.

Data Fields

`time_t ts`

`unsigned char value`

4.1.5 IotcCampaignCallbacks

IotcCampaignCallbacks represents a collection of campaign callback functions.

Data Fields

`IotcUpdateInventoryInfoCb * IotcCampaignCallbacks::inventoryInfoCb`

- An update inventory info callback function.

`IotcCampaignPreDownloadCb * IotcCampaignCallbacks::preDownloadCb`

- A pre-download callback function.

`IotcCampaignPreExecutionCb * IotcCampaignCallbacks::preExecutionCb`

- A pre-execution callback function.

`IotcCampaignExecuteCb * IotcCampaignCallbacks::executeCb`

- An execute callback function.

`IotcCampaignPreActivationCb * IotcCampaignCallbacks::preActivationCb`

- A pre-activate callback function.

`IotcCampaignActivateCb * IotcCampaignCallbacks::activateCb`

- An activate callback function.

`IotcCampaignDownloadProgressCb * IotcCampaignCallbacks::downloadProgressCb`

- A Download progress callback function.

`IotcCampaignStateChangeCb * IotcCampaignCallbacks::stateChangeCb`

- A state change callback function.

4.1.6 IotcCampaignId

`IotcCampaignIdIotcCampaignId` represents the campaign identifier. Campaign identifier is any string with a maximum length of `IOTC_UUID_SIZE - 1` . It is provided by the INFER server as a response to an agent API or server API, or through campaign callbacks. A campaign identifier could be in the GUID format such as `123e4567-e89b-12d3-a456-426655440000` , or any string such as `5b1656704cedfd000626bcaa`

Data Fields

`char id [IOTC_UUID_SIZE]`

- Holds the actual characters of the identifiers

4.1.7 IotcCampaignScheduleTimeWindow

`IotcCampaignScheduleTimeWindow` represents the campaign time window.

Contains the begin and end date and time for scheduling in the timestamp format expressed in epoch time. For example,

`beginTime - Stamp : 1534855979, endTimeStamp : 1534856979`

Data Fields

`time_t IotcCampaignScheduleTimeWindow::beginTimeStamp`

- Beginning timestamp for the time window.

`time_t IotcCampaignScheduleTimeWindow::endTimeStamp`

- Ending timestamp for the time window.

4.1.8 IotcClientConfig

`IotcClientConfig` represents client configuration SDKs.

Contains the begin and end date and time for scheduling in the timestamp format expressed in epoch time. For example,

`beginTime - Stamp : 1534855979, endTimeStamp : 1534856979`

Data Fields

`IotcApplicationId appId`

`IotcClientLogLevel logLevel`

4.1.9 IotcCommand

Command structure to hold details about a command message received from the server.

Data Fields

`char IotcCommand::name[IOTC_NAME_MAX_SIZE]`

- Friendly name of the command.

`char IotcCommand::id[IOTC_UUID_SIZE]`

- Command identifier generated by the server.

`IotcDeviceId IotcCommand::deviceId`

- Device identifier for the targeted device. This is an optional field.

`char IotcCommand::execPath[IOTC_PATH_MAX]`

- Absolute path to the executable. This is an optional field.

`size_t IotcCommand::numArgs`

- Number of command arguments.

`IotcCommandArg * args`

- List of arguments for the command.

4.1.10 IotcCommandArg

Command argument structure.

Data Fields

`IotcCommandArgValueType IotcCommandArg::type`

- The value type.

`char IotcCommandArg::name[IOTC_NAME_MAX_SIZE]`

- Name of the argument.

`IotcDeviceId IotcCommand::deviceId`

- Device identifier for the targeted device. This is an optional field.

`size_t IotcCommandArg::numValues`

- Number of items in the value array.

```
union
{
     int64_t * intValues
     double * doubleValues
     char ** strValues
};
```

- Value array of the argument.

4.1.11 IotcCommandResponse

Command response to hold the response received for a command.

Data Fields

`char IotcCommandResponse::message[IOTC_PAYLOAD_MAX_SIZE]`

- The error message to be sent to the server.

4.1.12 IotcDevice

(Deprecated) Represents a device entity.

Data Fields

`IotcDeviceId deviceId`

`IotcDeviceType type`

4.1.13 IotcDeviceDetails

Represents the device details.

Data Fields

`char IotcDeviceDetails::name[IOTC_NAME_MAX_SIZE]`

- Name of the device.

`char IotcDeviceDetails::deviceTemplate[IOTC_NAME_MAX_SIZE]`

- Name of the device template.

`char IotcDeviceDetails::deviceOrgId[IOTC_UUID_SIZE]`

- Organization ID for the Device

4.1.14 IotcDeviceId

IotcDeviceId represents the device identifier.

Data Fields

`char id [IOTC_UUID_SIZE]`

4.1.15 IotcDeviceData

IotcDeviceData represents a device entity.

Data Fields

`IotcDeviceId deviceId`

`IotcTemplateId templateId`

`IotcDeviceId parentId`

`IotcDeviceId parentGatewayId`

`IotcDeviceType type`

`IotcEnrollmentState enrollmentState`

`char deviceName [IOTC_NAME_MAX_SIZE]`

`char templateName [IOTC_NAME_MAX_SIZE]`

`IotcKeyValueSet systemProperties`

`IotcKeyValueSet customProperties`

`IotcAllowedMetricSet allowedMetrics`

4.1.16 IotcDeviceSet

Represents the set of devices.

Data Fields

`IotcDevice*device`

`size_t used`

`size_t size`

4.1.17 IotcDeviceDataSet

4.1.18 IotcDoubleValue

IotcDoubleValue represents the float type metric data point.

Data Fields

`time_t ts`

`double value`

4.1.19 IotcEnrollmentCredentials

IotcEnrollmentCredentials represents the enrollment credentials.

Data Fields

`char IotcEnrollmentCredentials::authToken[IOTC_PAYLOAD_MAX_SIZE]`

- authToken contains the credentials required by the enrollment provider.

4.1.20 IotcEnrollmentData

IotcEnrollmentData represents the enrollment data.Enrollment data contains the type of enrollment and the required data for the enrollment.

Data Fields

`IotcDeviceId IotcEnrollmentData::parentId`

- parentId is the device ID of the gateway device that the device connects to. For the root gateway device, the parent ID must be empty.

`IotcDeviceId IotcEnrollmentData::deviceId`

- deviceId must be set for the IOTC_PRE_REGISTERED type.

`IotcDeviceDetails IotcEnrollmentData::deviceDetails`

- deviceDetails must be set for the IOTC_NOT_REGISTERED type.

4.1.21 IotcEnrollmentRequest

IotcEnrollmentRequest represents the enrollment request structure.

Data Fields

`IotcEnrollmentData IotcEnrollmentRequest::data`

- Data contains the required enrollment data.

```
union
{
     IotcEnrollmentCredentials enrollmentCredentials
     IotcUserCredentials userCredentials
};
```

`IotcEnrollmentCredentials IotcEnrollmentRequest::enrollmentCredentials`

- enrollmentCredentials must be set for the IOTC_PRE_REGISTERED type.

`IotcUserCredentials IotcEnrollmentRequest::userCredentials`

- userCredentials must be set for the IOTC_NOT_REGISTERED type.

4.1.22 IotcEnrollmentResponse

IotcEnrollmentResponse represents the enrollment response.

Data Fields

`IotcDeviceId deviceId`

`IotcDeviceId parentId`

4.1.23 IotcGetResponse

IotcGetResponse represents the GetResponse sent from the agent to the SDK.

Data Fields

`uint64_t messageId`

`IotcGetResponseMsgType type`

`void * response`

4.1.24 IotcInt64Value

IotcInt64Value represents the integer type metric data point.

Data Fields

`time_t ts`

`int64_t value`

4.1.25 IotcKeyValue

IotcKeyValue represents a key value pair.

Data Fields

`char key [IOTC_NAME_MAX_SIZE]`

`char value [IOTC_VALUE_MAX_SIZE]`

4.1.26 IotcKeyValueSet

IotcKeyValueSet contains an array of device properties used. It represents the number of current keyValue set size and the capacity of the keyValue set.

Data Fields

`IotcKeyValue*keyValue`

`size_t used`

`size_t size`

4.1.27 IotcMetric

IotcMetric represents the metric data point to be sent to the agent.

Data Fields

`IotcMetricType type`

`IotcDeviceId deviceId`

`char name [IOTC_METRIC_NAME_SIZE]`

```
union
{
     struct IotcStringValue strings [0]
     struct IotcIntegerValue integers [0]
     struct IotcFloatValue floats [0]
     struct IotcBooleanValue bools [0]
};
```

4.1.28 IotcMetricResponse

IotcMetricResponse represents the metric response.

Data Fields

`IotcMetricResponseStatus`

- Metric status of type.

`metric`

- Metric information that is received from agent.

4.1.29 IotcNotificationDefinitionId

IotcNotificationDefinitionId represents the notification definition identifier.

Data Fields

`char IotcNotificationDefinitionId::id[IOTC_UUID_SIZE]`

- Holds the actual characters of the identifiers.

4.1.30 IotcNotificationResponse

IotcMetricIntvlResponse represents the notification response sent from the server to the client.

Data Fields

`IotcNotificationDefinitionId IotcNotificationResponse::definitionId`

- Notification definition identifier.

`IotcNotificationId IotcNotificationResponse::notificationId`

- Notification instance identifier.

`char IotcNotificationResponse::payload[IOTC_PAYLOAD_MAX_SIZE]`

- The payload byte size.

`int IotcNotificationResponse::status`

- Status flag.

4.1.31 IotcPackageId

`IotcPackageId` represents the package identifier string.The package identifier is a string with a maximum length of `IOTC_UUID_SIZE - 1` . It is in the GUID format such as, `98732222-1234-12d3-a456-426655440000` .

Data Fields

`char IotcPackageId::id[IOTC_UUID_SIZE]`

- Holds the actual characters of the identifiers.

4.1.32 IotcPropertySet

(Deprecated) IotcPropertySet represents information about the properties that are currently set.

Data Fields

`IotcDeviceId deviceId`

- Contains an array of devices used.

`IotcProperty * property`

- Contains an array of properties used.

`size_t used`

- Represents the number properties currently set.

`size_t size`

- Represents the capacity of the property set.

4.1.33 IotcSendNotificationRequest

IotcSendNotificationRequest represents the send notification request sent from the client to the server.

Data Fields

`IotcNotificationDefinitionId IotcSendNotificationRequest::definitionId`

- Notification definition identifier.

`IotcApplicationId IotcSendNotificationRequest::entityId`

- Source of the request.

`IotcKeyValue* IotcSendNotificationRequest::keyValues`

- Array of key value pairs.

`size_t IotcSendNotificationRequest::numKeyValues`

- Number of key value pairs.

4.1.34 IotcStringValue

IotcStringValue represents the string type metric data point.

Data Fields

`time_t ts`

`char value [IOTC_METRIC_STRING_VALUE_SIZE]`

4.1.35 IotcTemplateId

IotcTemplateId represents the template identifier.

Data Fields

`char id [IOTC_UUID_SIZE]`

4.1.36 IotcUploadFileRequest

IotcUploadFileRequest represents the Upload File request sent from the agent to the server.

Data Fields

`char IotcUploadFileRequest::srcFilePath[PATH_MAX]`

- File path at the local system to be uploaded.

`char IotcUploadFileRequest::dstFilePath[PATH_MAX]`

- Path with the destination file name appended at end of the URL to upload the file.

4.1.37 IotcUserCredentials

IotcUserCredentials represents basic user credentials and organization domain name.

Data Fields

`char username [IOTC_NAME_MAX_SIZE]`

`char password [IOTC_NAME_MAX_SIZE]`

`char orgDomainName [IOTC_NAME_MAX_SIZE]`

4.2 Functions

4.2.1 Iotc_AddMetricData

Adds metric data point in the metric data set.

API

```
int Iotc_AddMetricData (
     struct IotcMetricDataSet * metricDataSet,
     IotcMetric * metric )
```

Description

Sort the list based on device IDs. This ensures that all the metrics belonging to the same device are inserted from the device list when you fetch a device node.

Parameters

`metricDataSet[IN,OUT]`

- Is the pointer to the metric data set.

`metric`

- Is the metric data to be sent to Agent.

Returns

0 on success. -1 on failure.

4.2.2 Iotc_AllocatePropertySet

(Deprecated) Allocates memory for the property set to hold the size and the number of properties.

API

```
int Iotc_AllocatePropertySet ( IotcPropertySet * properties, size_t size )
```

Parameters

`in, out    properties`

- Pointer to the property set.

`in    size`

- Capacity of the property set in terms of number of properties.

Returns

0 on success and -1 on failure.

4.2.3 Iotc_AllocMetricDataSet

Allocates memory for metric data set to hold metrics data points

API

```
struct IotcMetricDataSet* Iotc_AllocMetricDataSet ( void )
```

Returns

Pointer to allocated metric data set structure on success, and NULL on failure.

4.2.4 Iotc_CampaignScheduleActivation

Schedules the campaign for activation.

API

```
int Iotc_CampaignScheduleActivation (
     IotcSession * session,
     IotcCampaignId * campaignId,
     IotcCampaignScheduleTimeWindow * timeWindow )
```

Description

Sends a request to the Server to schedule the campaign for activation. If the time window is empty, it indicates that client is ready to activate the campaign. Otherwise, the supplied time window is used by the server to schedule the campaign to activate for this gateway.

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

`in campaignId`

- Is the campaign ID of the campaign that is scheduled for activation.

`in timeWindow`

- Is the schedule time window for the campaign to activate.

Returns

0 on success.

4.2.5 Iotc_CampaignScheduleDownload

Schedules the campaign for download.

API

```
int Iotc_CampaignScheduleDownload (
     IotcSession * session,
     IotcCampaignId * campaignId,
     IotcCampaignScheduleTimeWindow * timeWindow )
```

Description

Sends a request to the server to schedule the campaign for download. If the time window is empty, it indicates that the client is ready for downloading the campaign. Otherwise, the supplied time window is used by the server to schedule download of the campaign for this gateway.

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

`in campaignId`

- Is the campaign ID of the campaign that is scheduled for download.

`in timeWindow`

- Is the schedule time window for the campaign to download.

Returns

0 on success.

4.2.6 Iotc_CampaignScheduleExecution

Schedules the campaign for execution.

API

```
int Iotc_CampaignScheduleExecution (
     IotcSession * session,
     IotcCampaignId * campaignId,
     IotcCampaignScheduleTimeWindow * timeWindow )
```

Description

Sends a request to the Server to schedule the campaign for running. If the time window is empty, it indicates that client is ready to run the campaign. Otherwise, the supplied time window is used by the server to schedule the campaign to run for this gateway.

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

`in campaignId`

- Is the campaign ID of the campaign that is scheduled for running.

`in timeWindow`

- Is the schedule time window for the campaign to run.

Returns

0 on success.

4.2.7 Iotc_CampaignSetExecutionProgress

Updates the execution progress of the campaign.

API

```
int Iotc_CampaignSetExecutionProgress (
     IotcSession * session,
     IotcCampaignId * campaignId,
     const char * progress )
```

Description

Sends a request to the server to update the execution progress of the campaign.

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

`in campaignId`

- Is the campaign identifier.

`in progress`

- Is progress string to be sent to the server.

Returns

0 on success.

4.2.8 IotcCommandCb

The command callback function type.

API

```
typedef int IotcCommandCb(
     const IotcCommand *command,
     IotcCommandResponse *response,
     void *context)
```

Parameters

`in`

- command is the command received from the server.

`out`

- response is the response data to be sent to the server for the command.

`in`

- context is the opaque context data that is supplied by the client during command callback registration.

Returns

0 on success. -1 on error.

4.2.9 Iotc_Close

Closes the communication channel with the IoTC Agent.

API

`void Iotc_Close ( IotcSession * session )`

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

4.2.10 Iotc_DeletePropertySet

(Deprecated) Frees the memory used by the property set.

API

`void Iotc_DeletePropertySet ( IotcPropertySet * properties )`

Parameters

`in, out properties`

- Pointer to the property set.

4.2.11 Iotc_DeleteProperties

Frees the memory used by the properties.

API

`void Iotc_DeleteProperties ( IotcKeyValueSet * properties )`

Parameters

`in, out properties`

- Pointer to the property set.

4.2.12 Iotc_Enroll

Enrolls the gateway and generates a Gateway Identifier.

API

```
int Iotc_Enroll (
     IotcSession * iotcSession,
     IotcEnrollmentRequest * enrollmentRequest )
```

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

`in requestData`

- Is the pointer to the enroll request object.

`out responseData`

- Contains the enroll response received for the request.

Returns

0 on success.

4.2.13 Iotc_FreeMetricDataSet

Frees the metric data points in the metric data set. Mandatory if Iotc_AllocMetricDataSet() is called.

API

```
void Iotc_FreeMetricDataSet ( struct IotcMetricDataSet * metricDataSet )
```

Parameters

`metricDataSet[IN]`

- Is the pointer to the metric data set.

Returns

0 on success. -1 on failure.

4.2.14 Iotc_GetCertificateIds

This function retrieves all the certificate Ids of the associated devices.

API

```
int Iotc_GetCertificateIds(
     IotcSession *iotcSession,
     const IotcDeviceId *deviceId )
```

Description

Get all the certificate Ids associated with a device.

Parameters

`iotcSession`

- The current IotcSession to be used.

`deviceId`

- Pointer to the device identifier of the device.

Returns

0 on success and -1 on failure.

4.2.15 Iotc_GetCertificateIdsByIssuer

This function retrieves all the certificate Ids associated with a devices matching the specified issuer.

API

```
int Iotc_GetCertificateIdsByIssuer(
     IotcSession *iotcSession,
     const IotcDeviceId *deviceId,
     const char *issuer )
```

Description

Get certificate ids associated with a device whose issuer entry matches the specified issuer.

Parameters

`iotcSession`

- The current IotcSession to be used.

`deviceId`

- Pointer to the device identifier of the device.

`issuer`

- NULL terminated UTF-8 string. This string is matched against all the entries in a certificate issuer(for example, Common Name, Organization Name). If one of the entries completely matches this string, the certificate Id is included in the response.

Returns

0 on success and -1 on failure.

4.2.16 Iotc_GetCertificateIdsBySubject

This function retrieves all the certificate Ids of the associated devices matching the subject.

API

```
int Iotc_GetCertificateIdsBySubject(
     IotcSession *iotcSession,
     const IotcDeviceId *deviceId,
     const char *subject)
```

Description

Get certificate ids associated with a device whose subject entry matches the appropriate subject.

Parameters

`iotcSession`

- The current IotcSession to be used.

`deviceId`

- Pointer to the device identifier of the device.

`subject`

- NULL terminated UTF-8 string. This string is matched against all the entries in a certificate subject(for example, Common Name, Organization Name). If one of the entries fully matches this string, the certificate id is included in the response.

Returns

0 on success and -1 on failure

4.2.17 Iotc_GetCertificate

This function retrieves the certificate associated with a device.

API

```
int Iotc_GetCertificate(
     IotcSession *iotcSession,
     const IotcDeviceId *deviceId,
     const IotcCertificateId *certId,
     const char *filePath)
```

Description

Get private key as a PEM file. The PEM file is written to a specified file path.

Parameters

`iotcSession`

- The current IotcSession to be used.

`deviceId`

- Pointer to the device identifier of the device.

`certId`

- Pointer to IotcCertificateId.

`filePath`

- The path to which the certificate is written. Must be a full path.

Returns

0 on success and -1 on failure

4.2.18 Iotc_GetPrivateKey

This function retrieves the private key.

API

```
int Iotc_GetPrivateKey(
     IotcSession *iotcSession,
     const IotcDeviceId *deviceId,
     const IotcCertificateId *certId,
     const char *filePath)
```

Description

Get private key as a PEM file. The PEM file is written to a specified file path.

Parameters

`iotcSession`

- The current IotcSession to be used.

`deviceId`

- Pointer to the device identifier of the device.

`certId`

- Pointer to IotcCertificateId

`filePath`

- The path to which the certificate is written. Must be a full path.

Returns

0 on success and -1 on failure

4.2.19 Iotc_GetCommands

Gets commands available for this gateway device from the Server.

API

```
int Iotc_GetData (
     IotcSession * session,
     IotcGetDataRequest * requestData )
```

Description

Sends a request to the server to check if there are any commands available for this gateway. If the retrieved command data is for the agent, then the agent processes it. Any command data that is not for the agent is returned to the client as response data.

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

Returns

0 on success and -1 on failure.

4.2.20 Iotc_GetCustomProperties

(Deprecated) Retrieves the custom properties of the gateway device.

API

```
int Iotc_GetCustomProperties (
     IotcSession * iotcSession,
     IotcDeviceId * deviceId )
```

Parameters

`in iotcSession`

- Is the connected session returned as part of Iotc_Init call.

`in deviceId`

- Is the device identifier.

Returns

0 on success and -1 on failure.

4.2.21 Iotc_FreeGetResponse

A general function to free internal resources used in a IotcGetResponse message.

API

```
void Iotc_FreeGetResponse ( IotcGetResponse * getResponse )
```

Parameters

`getResponse`

- pointer of the IotcGetResponse message.

4.2.22 Iotc_GetResponseByType

Processes response messages and returns only the desired message based on the message type provided.

API

```
int Iotc_GetResponseByType (
     IotcSession * session,
     IotcGetResponseMsgType requestedType,
     int timeout,
     IotcGetResponse * getResponse )
```

Parameters

`in  session`

- The current IotcSession to be used.

`in  requestedType`

- The desired type of response message to be obtained.

`in  timeout`

- The duration to wait for a response from the agent, in milliseconds.

`out  getResponse`

- IotcGetResponse pointer for holding the result.

Returns

Returns -1 on failure. This value comes from the status of the response message or from a communication error. To handle differences between these failures, check the returned message type.

4.2.23 Iotc_Sync

This function synchronizes device related information such as default properties with the server.

API

```
int Iotc_Sync ( IotcSession * iotcSession )
```

4.2.24 Iotc_GetDevices

(Deprecated) This function retrieves all the connected devices for a device using the ID and type.

API

```
int Iotc_GetDevices ( IotcSession * iotcSession, IotcDeviceId * parentId )
```

Description

The devices would be returned with a pointer to IotcDeviceSet in the IotcGetResponse with IOTC_GET_DEVICES as the response message type.

Parameters

`iotcSession`

- The current IotcSession to be used.

`parentId`

- Device ID for which the connected device IDs must be retrieved.

4.2.25 Iotc_GetDevicesData

This function retrieves details such as device ID, device type, device name, template ID, template name, enrollment state, parent ID, parent gateway ID, system properties, custom properties and allowed metrics of all the connected devices of a device.

API

```
int Iotc_GetDevicesData (
     IotcSession * iotcSession,
     IotcDeviceId * parentId )
```

Description

The devices are returned with a pointer to IotcDeviceDataSet in the IotcGetResponse with IOTC_GET_DEVICES_DATA as the response message type.

Parameters

`iotcSession`

- The current IotcSession to be used.

`parentId`

- Device ID for which the connected device IDs must be retrieved.

4.2.26 Iotc_GetMessageId

Returns the messageId corresponding to the latest API invoked by the client. Invoke Iotc_GetMessageId before calling the next API.

API

```
uint64_t Iotc_GetMessageId ( IotcSession * iotcSession )
```

Parameters

`in  iotcSession`

- Is the connected session returned as part of Iotc_Init call.

4.2.27 Iotc_GetResponse

Gets response from the agent.

API

```
int Iotc_GetResponse (
     IotcSession * iotcSession,
     IotcGetResponse * response )
```

Description

Returns GetData response to the Client. If the retrieved command data is for the agent, then agent processes it. Any command data that is not for the agent is returned to the client as response data.

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

`out responseData`

- Contains the response data received for the request.

Returns

0 on success and -1 on failure.

4.2.28 Iotc_GetSessionSockfd

(Deprecated) Retrieves the system properties of the gateway device.

API

```
int Iotc_GetSystemProperties (
     IotcSession * iotcSession,
     IotcDeviceId * deviceId )
```

Parameters

`in iotcSession`

- Is the connected session returned as part of Iotc_Init call.

`in deviceId`

- Is the device identifier.

Returns

0 on success and -1 on failure.

4.2.29 IotcSession* Iotc_Init

Initializes the communication channel with the IoTC Agent.

API

`IotcSession* Iotc_Init ( IotcApplicationId * applicationId )`

Parameters

`in applicationId`

- Is the application identifier of the invoking client.

Returns

Pointer to the session object on success or NULL on failure.

4.2.30 IotcSession*Iotc_InitWithConfig

Initializes a communication channel with the IoTC Agent using the supplied configuration.

API

`IotcSession*Iotc_InitWithConfig ( IotcClientConfig * config )`

Parameters

`in config`

- Is a pointer to the client configuration object.

Returns

Pointer to the session object on success or NULL on failure.

4.2.31 Iotc_InsertProperty

(Deprecated) Adds a property to the property set.

API

```
int Iotc_InsertProperty (
     IotcPropertySet * properties,
     IotcProperty * property )
```

Parameters

`in, out properties`

- Pointer to the property set.

`in property`

- Pointer to the property to be added.

Returns

0 on success and -1 on failure.

4.2.32 Iotc_InsertProperties

Adds a property to the property set.

API

```
int Iotc_InsertProperties (
     IotcKeyValueSet * properties,
     IotcKeyValue * property )
```

Parameters

`in, out properties`

- Pointer to the property set.

`in property`

- Pointer to the property to be added.

Returns

0 on success and -1 on failure.

4.2.33 Iotc_RegisterCampaignCallbacks

Registers campaign callback functions.

API

```
int Iotc_RegisterCampaignCallbacks (
     IotcSession * session,
     IotcCampaignCallbacks * cbs,
     void * userData )
```

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

`in cbs`

- Is the campaign callback functions collection that is invoked by the IoTC Agent during state change, download progress, download, and so on.

`in userData`

- Is any user context data that is returned when invoking callback functions.

Returns

0 on success.

4.2.34 Iotc_RegisterCommandCallback

The command callback registration function.

API

```
int Iotc_RegisterCommandCallback (
     IotcSession * iotcSession,
     IotcCommandCb * cb,
     void * context )
```

Parameters

`in IotcSession`

- The current IoTCSession to be used.

`in cb`

- The command callback function.

`in context`

- The pointer to any context data that must be supplied when cb is called.

Returns

0 on success. -1 on error.

4.2.35 Iotc_SendMetric

Requests the agent to send a metric to the server.

API

```
int Iotc_SendMetric ( IotcSession * iotcSession, IotcMetric * requestData )
```

Parameters

`in iotcSession`

- Is the connected session returned as part of Iotc_Init call.

`in requestData`

- Is the pointer to the send metric request data.

4.2.36 Iotc_SendMetricSet

Sends multiple metrics to the Agent to be sent to the server.

API

```
int Iotc_SendMetricSet (
     IotcSession * iotcSession,
     struct IotcMetricDataSet * metricDataSet )
```

Description

Use following helper functions to add metrics data:

```
Iotc_MetricDataSet *Iotc_AllocMetricDataSet(void);
```

```
Iotc_AddMetricData(IotcMetricDataSet *metricDataSet, IotcMetric *metric);
```

4.2.37 Iotc_SendNotification

Sends the notification request to the server.

API

```
IotcSession * session, IotcSendNotificationRequest * requestData
```

Parameters

`in session`

- Is the connected session returned from Iotc_Init call.

`in requestData`

- Is the pointer to the notification request object.

Returns

0 on success.

4.2.38 Iotc_SendPropertySet

Sends the property set to the server.

API

```
int Iotc_SendPropertySet (
     IotcSession * iotcSession,
     IotcKeyValueSet * properties,
     IotcDeviceId * deviceId )
```

Parameters

`in iotcSession`

- Is the connected session returned from Iotc_Init call.

`in properties`

- Pointer to the property set.

`in deviceId`

- Device identifier.

Returns

0 on success and -1 on failure.

4.2.39 Iotc_UnEnroll

Requests to un-enroll a device.

API

```
int Iotc_UnEnroll ( IotcSession * iotcSession, IotcDeviceId * deviceId )
```

Description

Sends a request to the Server to un-enroll the device specified by deviceId. If the deviceId is empty, then the root gateway device is un-enrolled.

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

`in deviceId`

- Pointer to the device identifier of the device.

4.2.40 Iotc_UploadFile

Uploads the specified file to the server.

API

```
int Iotc_UploadFile (
     IotcSession * session,
     IotcUploadFileRequest * requestData )
```

Parameters

`in session`

- Is the connected session returned as part of Iotc_Init call.

`in requestData`

- Is the pointer to the post data request object.

Returns

0 on success.

4.3 Macro Definitions

This section lists the macros and their definitions for the Agent APIs.

`#define IOTC_UUID_SIZE 37`

- The maximum size of the UUID.

`#define IOTC_NAME_MAX_SIZE 256`

- The maximum size for a name string. Used in device names and in device property name-value pairs.

`#define IOTC_VALUE_MAX_SIZE 512`

- The maximum size for a value string. Used in device property name-value pairs.

`#define IOTC_APP_ID_SIZE 65`

- The maximum size of an application identifier.

`#define IOTC_PAYLOAD_MAX_SIZE 4096`

- The maximum size for the payloads.

```
#define IOTC_METRIC_NAME_SIZE 64
```

- The maximum size of the metric name.

```
#define IOTC_METRIC_STRING_VALUE_SIZE 32
```

- The maximum size of the metric string data point.

4.4 Enumeration Types

This section lists the enumeration types and their definitions for the Agent APIs.

4.4.1 enum IotcValType

Denotes the metric unit type.

```
{
     BOOLEAN,
     FLOAT,
     STRING,
     INTEGER
}
```

4.4.2 enum IotcCampaignState

Denotes the supported campaign states.

```
{
     IOTC_CAMPAIGN_INITIALIZED,
     IOTC_CAMPAIGN_INSTANTIATED,
     IOTC_CAMPAIGN_INVENTORY_UP_TO_DATE,
     IOTC_CAMPAIGN_INVENTORY_UPDATE_FAILURE,
     IOTC_CAMPAIGN_WAITING_FOR_DOWNLOAD_APPROVAL,
     IOTC_CAMPAIGN_SCHEDULED_DOWNLOAD,
     IOTC_CAMPAIGN_WAITING_FOR_DOWNLOAD,
     IOTC_CAMPAIGN_DOWNLOADING,
     IOTC_CAMPAIGN_DOWNLOAD_COMPLETE,
     IOTC_CAMPAIGN_DOWNLOAD_FAILED,
     IOTC_CAMPAIGN_WAITING_FOR_EXECUTION_APPROVAL,
     IOTC_CAMPAIGN_SCHEDULED_EXECUTION, IOTC_CAMPAIGN_WAITING_TO_EXECUTE,
     IOTC_CAMPAIGN_EXECUTING,
     IOTC_CAMPAIGN_EXECUTION_COMPLETE,
     IOTC_CAMPAIGN_EXECUTION_FAILED,
     IOTC_CAMPAIGN_WAITING_FOR_ACTIVATION_APPROVAL,
     IOTC_CAMPAIGN_SCHEDULED_ACTIVATION,
     IOTC_CAMPAIGN_WAITING_TO_ACTIVATE,
     IOTC_CAMPAIGN_ACTIVATING,
     IOTC_CAMPAIGN_ACTIVATION_COMPLETE,
     IOTC_CAMPAIGN_ACTIVATION_FAILED
}
```

4.4.3 enum IotcGetResponseMsgType

Denotes the supported response message types.

```
{
     IOTC_INVALID_RESPONSE,
     IOTC_NOTIFICATION_RESPONSE,
```

```
    IOTC_ENROLL_RESPONSE,
    IOTC_UNENROLL_RESPONSE,
    IOTC_CAMPAIGN_STATE_CHANGE,
    IOTC_SCHEDULE_RESPONSE,
    IOTC_SET_PROGRESS,
    IOTC_SEND_METRIC,
    IOTC_UPLOAD_FILE,
    IOTC_GET_COMMANDS_FINISHED,
    IOTC_REGISTER_CB,
    IOTC_SEND_PROPERTIES,
    IOTC_GET_SYSTEM_PROPERTIES,
    IOTC_GET_CUSTOM_PROPERTIES,
    IOTC_GET_DEVICES,
    IOTC_GET_DEVICES_DATA,
    IOTC_CLIENT_COMMAND,
    IOTC_ERROR_RESPONSE,
    IOTC_NO_RESPONSE
}
```

4.4.4 enum IotcEnrollmentType

Denotes the supported enrollment types.

```
{
    IOTC_PRE_REGISTERED,
    IOTC_NOT_REGISTERED
}
```

4.4.5 enum boolean

Denotes the boolean state.

```
{
    FALSE,
    TRUE
}
```

4.4.6 enum IotcMetricType

Denotes the metric unit type.

```
{
    IOTC_METRIC_ERROR,
    IOTC_METRIC_STRING,
    IOTC_METRIC_INTEGER,
    IOTC_METRIC_FLOAT,
    IOTC_METRIC_BOOLEAN,
    IOTC_METRIC_UNKNOWN
}
```

4.4.7 enum IotcMetricResponseStatus

Status of the metric response sent from the Agent SDK.

- IOTC_METRIC_FAILED: Metric failed to be stored at the agent or sent to the server.
- IOTC_METRIC_NOT_ALLOWED: The metric is not in the allowed list.
- IOTC_METRIC_STORED: The metric is successfully stored in the agent.

- IOTC_METRIC_SUCCESS: The metric is successfully sent to the server.

```
{
     IOTC_METRIC_SUCCESS,
     IOTC_METRIC_STORED,
     IOTC_METRIC_NOT_ALLOWED,
     IOTC_METRIC_FAILED
}
```

4.4.8 IotcClientLogLevel

Denotes the SDK client log levels.

```
{
     IOTC_LOG_EMERG = 0,
     IOTC_LOG_ALERT = 1,
     IOTC_LOG_CRIT = 2,
     IOTC_LOG_ERROR = 3,
     IOTC_LOG_WARN = 4,
     IOTC_LOG_NOTICE = 5,
     IOTC_LOG_INFO = 6,
     IOTC_LOG_DEBUG = 7
}
```

4.4.9 enum IotcEnrollmentState

Supported device states.