Workgroup: Network Working Group
Internet-Draft: draft-brinckman-nipc-00
Published: 20 October 2023
Intended Status: Standards Track
Expires: 22 April 2024
Authors: B. Brinckman R. Mohan B. Sanford
Cisco Systems Cisco Systems Philips

An Application Layer Interface for Non-IP device control (NIPC)

Abstract

This memo specifies RESTful application layer interface for gateways providing operations against non-IP devices. The described interface is extensible. This memo initially describes Bluetooth Low Energy and Zigbee as they are the most commonly deployed.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 April 2024.

Table of Contents

1. Introduction

Use cases in building management, healthcare, workplaces, manufacturing, logistics and hospitality have introduced low-power devices into these environments. These devices typically do not support IP-based interfaces, hence there is a need for gateway functions to allow these devices to communicate with the applications that manage them.

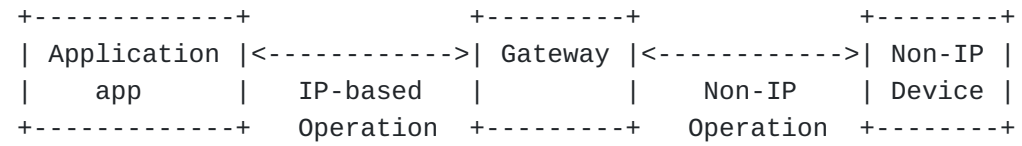

Figure 1: Gateway for non-IP Devices

In abscence of a standard describing how applications communicate with such non-IP devices, vertically integrated infrastructure prolifilates and applications have bespoke integrations with that infrastructure for every use case. The Application interfaces are non-standard. This stunts the eco-system growth. At the same time, wireless access points have been deployed nearly everywhere, many of which have soft or separate radios that can transmit and receive different frame types, such as [BLE53] and [Zigbee22]. To avoid the need for parallel infrastructure and bespoke application integration, a standardized gateway function is necessary.

The gateway provides at a minimum the following functions:

* authentication and authorization of application clients that will access devices

* the ability to onboard devices that are intended to be deployed within the use case

* maintenance of an inventory of onboarded devices that are intended to access and be accessed by the deployment and applications.

* interfaces that allow for bi-directional communication to non-IP devices

* one or more channels to process requests, responses, and asymmetric communciations with the non-IP radio resources (Access Points) in the system.

Combined with a provisioning interface such as [I-D.shahzad-scim-device-model], this specification supports these aspects, specifically focusing on providing bi-directional communication with non-IP devices.

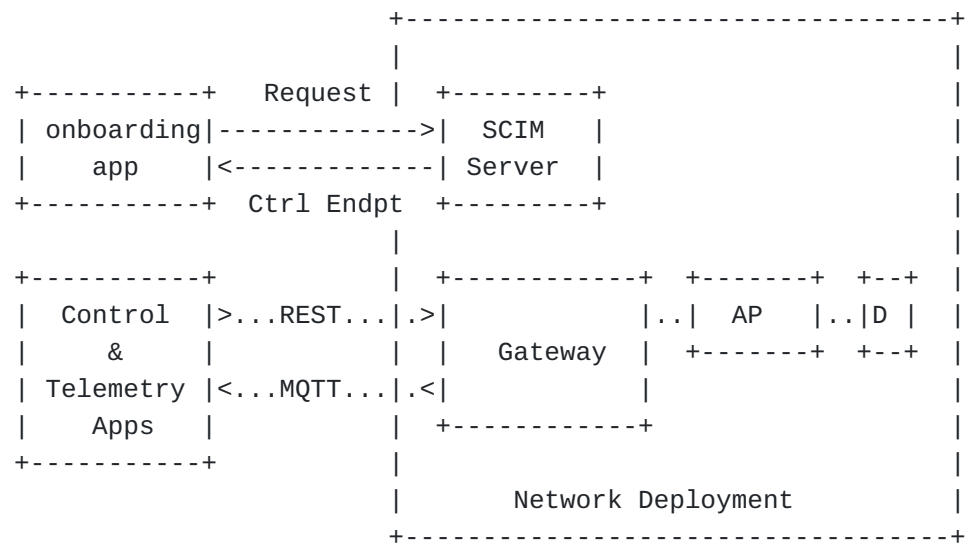

Figure 2: Basic Architecture

Figure 2 shows us the application layer gateway (ALG), an access point (AP), and a device (D) in the enterprise environment. The role of the ALG is to provide a application gateway to non-IP devices connecting into one or more AP. Applications implementing this memo can leverage RESTful interfaces to communicate with these devices and subscribe to streaming data or broadcasts levering MQTT.

The flow of operations are as follows:

1. The operator of the network deployment authorizes application(s) to perform operations on the Gateway. This happens out of band and may be accomplished by means of exchanging tokens or public keys. Authorization can be role-based:
   a. Authorize an onboarding application against a SCIM endpoint supported by the gateway.
   b. Provision and authorize applications that may control devices.
   c. Provision and authorize applications that may receive telemetry.

2. The authorized application can now provision one or more devices on the gateway leveraging SCIM.

Steps 1 and 2 are not within the scope of this specification, but are provided for context.

1. The authorized application can perform RESTful calls to the gateway in order to establish bi-directional communication to one or more devices. Optionally, set up a publish/subcribe topic to receive streaming data from a device (telemetry interface).

2. Optionally, an application can receive streaming data on a pub/sub topic configured by the control interface (telemetry interface).

Step 3 and 4 are the subject of this memo.

This specification is organized into three sections:

*Basic non-IP control functions described in narrative.

*Extensibility of the interfaces.

*A specification that can be mapped to a publication/subscribe interface, such as MQTT.

*Examples of use cases leveraging both BLE and Zigbee-based devices.

*OpenAPI definitions for the control interface and Protobuf definitions for the streaming data interface

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Non-IP Control Functions

This section will describe a standardized protocol-agnostic interface that allows the application to establish bi-directional communication with a non-IP device, such as a BLE or Zigbee device. The interface will be supported on a gateway as show in Figure 2.

2.1. Approach

In non-IP protocols such as BLE or Zigbee, a number of basic operations are defined that are similar across protocols. Examples of this are read and write data. Devices may choose to implement all

of the operations or a subset. For example in BLE a device may choose to implement a binding, but could also allow connection without a binding. In this memo we have therefore defined a control interface that exposes these basic operations with a communications protocol-agnostic schema, with protocol specific extensions to transmit and receive attributes that are specific to the communications protocol supported by the device. This enables extensions to integrate new non-ip communications protocols, without the need to update the base schema.

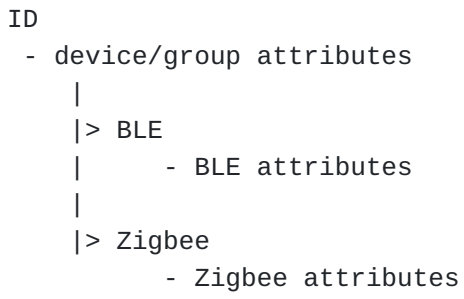

Figure 3: Extensible Schema

As shown in Figure 3, the control interface addresses device and group objects as IDs, hence the requirement to declare a device to the gateway before addressing a NIPC operation to the device. This is done by means of SCIM. A NIPC operation can either be performed against a device-id or a group-id. The gateway will leverage information from the SCIM object to execute a specific NIPC operation. For example, keying material found in the SCIM object may be required to connect to a device. Please refer to [I-D.shahzad-scim-device-model] for more information on SCIM device objects.

Apart from enabling bi-directional communication with non-ip devices, NIPC also allows an application to register pub/sub topics in order to support a programmable data streaming interface.

2.1.1. Common base schema

As described, most operations are executed against a device or a group. Control operations refer to either of these as "Object" with an ID as an identifier. The common schema for Object is defined as follows:

Attribute	Req	Type	Example
id	T	uuid	12345678-1234-5678-1234-56789abcdef4
type	T	enum	device

Attribute	Req	Type	Example
technology	F	enum	ble

Table 1: Definition of an Object

where-

*id is the id returned in the response when registering a device against a SCIM server.

*type is either "group" or "device".

*technology is the radio technology extension(s) supported by the device, in this memo either "ble" or "zigbee".

2.1.2. Protocol extensions

An object can support one or more communications protocols. These attributes must be described in a protocol object, for example a "ble" or a "zigbee" object.

Attribute	Req	Type	Example
ble	T	object	an object with BLE-specific attributes
zigbee	T	object	an object with Zigbee-specific attributes

Table 2: Protocol extensions

where-

*"ble" is an object containing attributes that are specific to the BLE protocol.

*"zigbee" is an object containing attributes that are specific to the Zigbee protocol.

*Other protocol extensions can be added

2.1.3. Response

As most operations have a common base schema, so do responses. As mandatory, a status is returned, optionally also device id and request id.

Success response:

Attribute	Req	Type	Example
status	T	enum	SUCCESS
id	F	uuid	12345678-1234-5678-1234-56789abcdef4
requestID	F	uuid	abcd0987-1234-5678-1234-56789abcdef4

Table 3: Success response

Failure response:

Attribute	Req	Type	Example
status	T	enum	SUCCESS
errorCode	T	int	12
reason	T	string	"Not Found"
requestID	F	uuid	abcd0987-1234-5678-1234-56789abcdef4

Table 4: Failure response

where-

*status is the status of the request, either "SUCCESS" or "FAILURE". In case of failure an error code and reason are added

*id is the id the operation was executed against, found in the request

*requestID is a correlation ID that can be used for end-to-end tracing.

*errorCode is a numerical value representing the error

*reason is a human readable explanation of why the error occurred

2.1.4. Categories of operations supported

The common operations are categorized in common categories that describe high level sets of functionalities. Each of the NIPC operations belong to a category. The categories are:

*/connectivity: Allows an application to establish connectivity with a device (if so required by the technology)

*/data: Allows applications to exchange data with a device

*/registrations: Allows an application to make registrations in the network, for example to register a pub/sub topic

*/extensions: This is a category of operations that leverage basic connectivity, data or registration operations, but are optimized for application usage, allowing applications to perform functions with a reduced number of round-trips. An example of this is the the bulk operation, allowing to send multiple operations is one operation. This category also allows for further extensions based on the basic operations.

2.1.5. Connecting to the Non-IP Control Interface

NIPC makes use of RESTful HTTP[RFC9114]. The connection endpoint is provided out of band, most likely through the SCIM devices model extension, in which an authorized application can be registered for a SCIM object. Similarly authentication of the interface can be specified using that SCIM interface. It may be based on a device certificate or an authorization token.

2.2. Connectivity

/connectivity

Connectivity elements are elements that allow operations that establish or tear down associations & connectivity with devices. They also allow discovery of services that can be accessed during the connection.

2.2.1. Binding

/connectivity/binding

The binding element allows an application to request a binding or association to a device.

Operations:

- *Create binding: POST
- *Return active bindings: GET
- *Delete binding: DELETE

2.2.1.1. Create a Binding

Method: POST /connectivity/binding

Description: Creates a binding with a device

Parameters: None

Request Body: an Object as defined in Table 1

Response: See Table 3 for success, and Table 4 for failed responses.

2.2.1.2. Return active bindings

Method: GET /connectivity/binding

Description: Returns one or more bindings, based on ids provided in parameters (none = return all)

Parameters: One of following options: - None: return all bindings this application made - single id: return binding for this id - comma separated ids: return bindings for multiple ids

Response: An Array of bindings with contents as shown in Table 5 below or Table 4 for failed responses.

Attribute	Req	Type	Example
status	T	enum	SUCCESS
requestID	F	uuid	abcd0987-1234-5678-1234-56789abcdef4
bindings	T	array	Array of BLE or Zigbee ids

Table 5: Binding response

2.2.1.3. Delete a binding

Method: DELETE /connectivity/binding

Description: Delete one or more bindings, based on ids provided in parameters

Parameters: One of following options: - None: delete all bindings this application made - single id: delete binding for this id - comma separated ids: delete bindings for multiple ids

Response: See Table 3 for success, and Table 4 for failed responses.

2.2.2. Binding by id

/connectivity/binding//id/{id}

The binding by id element allows an application to request a binding or association to a device by id, which provides a simpler interface than standard binding element, but pertains to a single device only.

Operations:

*Create binding by id: POST

*Return active binding by id: GET

*Delete binding by id: DELETE

2.2.2.1. Create a Binding by id

Method: POST /connectivity/binding/id/{id}

Description: Creates a binding by id

Parameters: id

Response: See Table 3 for success, and Table 4 for failed responses.

2.2.2.2. Return active binding by id

Method: GET /connectivity/binding//id/{id}

Description: Returns a binding by id Parameters: id

Response: See Table 3 for success, and Table 4 for failed responses.

2.2.2.3. Delete binding by id

Method: DELETE /connectivity/binding/id/{id}

Description: Delete a binding by id

Parameters: id

Response: See Table 3 for success, and Table 4 for failed responses.

2.2.3. Connection

/connectivity/connection

The connection element allows an application to request to connect to a device.

Operations:

- *Connect to a device: POST
- *Return active connections: GET
- *Disconnect a device: DELETE

2.2.3.1. Connect to a device

Method: POST /connectivity/connection

Description: Connect to a device

Parameters: None

Request Body: - an Object, as defined in Table 1 - optionally a set of services to be discovered. These are supplied in protocol-specific extensions, as defined in Table 2. In the case of BLE, service discovery is performed when connecting to a device. Optionally, service discovery may be limited to services defined in the "ble" protocol extension. The services to be discovered can be added in an array, as well as optional caching parameters. Please

see table below Table 6 for a definition of the content of the BLE protocol extension for limited service discovery

Attribute	Req	Type	Example
services	T	array	Array of serviceIDs to be discovered
cached	F	boolean	no
cacheIdlePurge	F	int	3600
autoUpdate	F	boolean	yes

Table 6: Service Discovery

where-

*"services" is an array of services defined by their serviceIDs.

*"cached" refers to whether the services need to be cached for subsequent connects, in order not to perform service discovery on each request.

*"cacheIdlepurge" defines how long the cache should be maintained before purging

*some devices support notifications on changes in services, "autoUpdate" allows the network to update services based on notification (on by default)

Response: Success responses include standard success response attributes as defined in Table 3 and also include cwan array of supported services. This array of supported services in turn contains an array of charateristics, which in turn contains an array of descriptors, as shown Figure 4. For a description of the attributes found in this array, please refer to Table 7 below. Please refer to Table 4 for failed responses.

```
 services
  - serviceID
     |
     |> characteristics
         - charactericID
         - flags
            |
            |> Descriptors
                - descriptorID
```

Figure 4: Services

Attributes in the array of services:

Attribute	Req	Type	Example
serviceID	F	uuid	abcd0987-1234-5678-1234-56789abcdef4
characteristicID	F	uuid	abcd0987-1234-5678-1234-56789abcdef4
flags	F	enum	write
descriptorID	F	uuid	abcd0987-1234-5678-1234-56789abcdef4

Table 7: Service Discovery Response Attributes

2.2.3.2. Return active connections

Method: GET /connectivity/connection

Description: Returns one or more active connections, based on ids provided in parameters (none = return all).

Parameters: One of following options: - None: return all active connections for this application - single id: return connection status for this id - comma separated ids: return connection status for multiple ids

Response: An Array of connections with attributes as defined in Table 8 or in case of a failed response, the attributes in Table 4.

Attribute	Req	Type	Example
status	T	enum	SUCCESS
requestID	F	uuid	abcd0987-1234-5678-1234-56789abcdef4
connections	T	array	Array of connections

Table 8: Connection response

2.2.3.3. Disconnect a device

Method: DELETE /connectivity/connection

Description: Disconnect one or more devices, based on ids provided in parameters

Parameters: One of following options: - None: Disconnect all devices for connections this application made - single id: disconnect device with id - comma separated ids: disconnect multiple devices with ids

Response: See Table 3 for success, and Table 4 for failed responses.

2.2.4. Connections by id

/connectivity/connection/id/{id}

The connection by id element allows an application to request a connection to a device by id, which provides a simpler interface than standard connection element, but pertains to a single device only.

Operations:

*Connect device by id: POST

*Return connection state by id: GET

*Disconnect device by id: DELETE

2.2.4.1. Connect device by id

Method: POST /connectivity/connection/id/{id}

Description: Creates a connection by id

Parameters: id

Response: See Table 3 for success, and Table 4 for failed responses.

2.2.4.2. Return connection state by id

Method: GET /connectivity/connection/id/{id}

Description: Returns connection state by id Parameters: id

Response: See Table 3 for success, and Table 4 for failed responses.

2.2.4.3. Delete connection by id

Method: DELETE /connectivity/connection/id/{id}

Description: Delete a binding by id

Parameters: id

Response: See Table 3 for success, and Table 4 for failed responses.

2.2.5. Discover services supported by a device

/connectivity/services

The services element allows an application to request a service discovery for a device, possibly to update a cache

Operations:

*Discover services: GET

2.2.5.1. Discover services supported by a device

Method: GET /connectivity/service

Description: Discover services supported by a device, this updates cache in case services caching is enabled for a connection.

Parameters: an Object as defined in Table 1 and optionally a set of services to be discovered in case not all services should be discovered. These services need to be provided in protocol-specific extensions as defined in Table 2. The services to be discovered can be added in this extension in an array, as well as optional caching parameters, as described in Table 6.

Response: A successful response will contain success attributes from Table 3 with an array of supported services characteristics and descriptors, as shown in Figure 4, with attributes defined in Table 7. Failure Response is the standard response as defined in Table 4

2.2.6. Discover services supported by a device by id

/connectivity/services/id/{id}

The services element allows an application to request a service discovery for a device by id, possibly to update a cache

Operations:

*Discover services by id: GET

2.2.6.1. Discover services supported by a device by id

Method: GET /connectivity/service/id/{id}

Description: Discover services supported by a device, this updates cache in case services caching is enabled for a connection. This method does not support partial service discovery, all services are discovered.

Parameters: an Object id

Response: A successful response will contain success attributes from Table 3 with an array of supported services, characteristics and descriptors, as shown in Figure 4, with attributes defined in Table 7. Failure Response is the standard response as defined in Table 4.

2.3. Data

/data

Data elements are elements that allow operations to exchange data with a device. This could be reading or writing attributes or enabling streaming data.

2.3.1. Attribute

/data/attribute

The attribute element allows an application get an attribute value, write, update or delete a value.

Operations:

*Write value: POST

*Update value: PUT

*Read value: GET

*Delete value: DELETE

2.3.1.1. Writing a value

Method: POST /data/attribute

Description: Writes a value to an attribute

Parameters: None

Request Body: an Object as defined in Table 1, a value to be written, as defined in Table 9 below and an attribute definition in a protocol extension from Table 2. The protocol extension for BLE is defined in Table 10 below. The protocol extension for Zigbee is defined in Table 11 below.

Attribute	Req	Type	Example
value	T	array	100
forcedResponse	F	boolean	no

Table 9: Value

where-

*value is the value to be written

*forcedresponse requests a specific response behavior of the device

Contents of the BLE protocol extension defining an attribute:

Attribute	Req	Type	Example
serviceID	T	uuid	abcd0987-1234-5678-1234-56789abcdef4
characteristicID	T	uuid	abcd0987-1234-5678-1234-56789abcdef4
long	F	boolean	no

Table 10: BLE Attribute

where-

*serviceID defines the Service

*characteristic ID defines the service characteristic

*long is an optional attribute that allows to force a write type

Contents of the Zigbee protocol extension defining an attribute:

	endpointID	T	int	16
	clusterID	T	int	6
	attributeID	T	int	12
	type	T	int	1

Table 11: Zigbee Attribute

where-

*endpointID defines the Zigbee endpoint that contains a cluster of attributes

*clusterID defines the Zigbee cluster that contains the attribute

*attributeID defines the Zigbee attribute

*type defines the Zigbee attribute type

Response: A successful response will contain success attributes from Table 3 with optionally the value written as shown in Table 9. Failure Response is the standard response as defined in Table 4

2.3.1.2. Updating a value

Method: PUT /data/attribute

Description: Updates a value to an attribute

Parameters: None

Request Body: an Object as defined in Table 1, a value to be written, as defined in Table 9 and an attribute definition in a protocol extension from Table 2. The protocol extension for BLE is

defined in Table 10. The protocol extension for Zigbee is defined in Table 11.

Response: A successful response will contain success attributes from Table 3 with optionally the value written as shown in Table 9. Failure Response is the standard response as defined in Table 4

2.3.1.3. Read an attribute

Method: GET /data/attribute

Description: Reads an attribute from a device

Parameters: an Object as defined in Table 1 and an attribute definition in a protocol extension from Table 2. The protocol extension for BLE is defined in Table 10. The protocol extension for Zigbee is defined in Table 11.

Response: A successful response will contain success attributes from Table 3 with the value read as shown in Table 9. Failure Response is the standard response as defined in Table 4

2.3.1.4. clear the value from an attribute

Method: DELETE /data/attribute

Description: Clear the value from an attribute of a device

Parameters: an Object as defined in Table 1 and an attribute definition in a protocol extension from Table 2. The protocol extension for BLE is defined in Table 10. The protocol extension for Zigbee is defined in Table 11.

Response: A successful response will contain success attributes from Table 3 with the optionally the value as null as shown in Table 9. Failure Response is the standard response as defined in Table 4

2.3.2. Subscription

/data/subscription

The subscription element allows an application to ask a device to start streaming data attached to a certain attribute.

Operations:

*Start a subscription data stream: POST

*Update a subscription data stream value: PUT

*Get status of a subscription data stream: GET

*Stop a subscription data stream: DELETE

2.3.2.1. Starting a subscription data stream

Method: POST /data/subscription

Description: Start a subcription data stream pertaining to a specific attribute

Parameters: None

Request Body: an Object as defined in Table 1 and an attribute definition in a protocol extension from Table 2. The protocol extension for BLE is defined in Table 10. The protocol extension for Zigbee is defined in Table 11. Optionally a pub/sub topic can be included in the request as defined in Table 12 below. Including a topic allows the app to skip the topic registration process.

Topic attributes:

Attribute	Req	Type	Example
topic	F	string	"enterprise/hospital/pulse"
dataFormat	F	enum	"default"
replay	F	boolean	no
forced_ack	F	boolean	no

Table 12: Topic

where-

*topic is the pub/sub topic the subscription can be consumed on

*dataFormat is the data format in which the pub/sub topic is delivered

*replay is a boolean which defines whether data should be replayed in case of application disconnection

*forced ack ignores the attribute definition and forces packet ack behavior to the device

Response: See Table 3 for success, and Table 4 for failed responses.

2.3.2.2. Updating a subscription data stream

Method: PUT /data/subscription

Description: Update parameters of a subscription data stream pertaining to a specific attribute

Parameters: None

Request Body: an Object as defined in Table 1 and an attribute definition in a protocol extension from Table 2. The protocol extension for BLE is defined in Table 10. The protocol extension for Zigbee is defined in Table 11. Optionally a pub/sub topic can be included in the request as defined in Table 12.

Response: See Table 3 for success, and Table 4 for failed responses.

2.3.2.3. Get status of a subscription data stream

Method: GET /data/subscription

Description: Gets the status of a subscription data stream, success if active, failure if not active

Parameters: an Object as defined in Table 1 and an attribute definition in a protocol extension from Table 2. The protocol extension for BLE is defined in Table 10. The protocol extension for Zigbee is defined in Table 11. Optionally a pub/sub topic can be included in the request as defined in Table 12.

Response: See Table 3 for success, and Table 4 for failed responses.

2.3.2.4. stop a subscription data stream

Method: DELETE /data/subscription

Description: stops a subscription data stream

Parameters: an Object as defined in Table 1 and an attribute definition in a protocol extension from Table 2. The protocol extension for BLE is defined in Table 10. The protocol extension for Zigbee is defined in Table 11. Optionally a pub/sub topic can be included in the request as defined in Table 12.

Response: See Table 3 for success, and Table 4 for failed responses.

2.3.3. Subscriptions by id

/data/subscription/id/{id}

The subscription by id element allows an application to operate on all subscriptions of a specific id.

Operations:

*Return active subscriptions by id: GET

*Terminate all subscriptions of an id: DELETE

2.3.3.1. Return active subscriptions by id

Method: GET /data/subscription/id/{id}

Description: Returns connection state by id

Parameters: id

Response: Success response as defines in Table 3 and an object called "subscriptions" which contains an Array of active subscriptions. Each of these include an object as defined in Table 1 and a subscription attribute definition in a protocolextension from Table 2. The protocol extension for BLE is defined in Table 10. The protocol extension for Zigbee is defined in Table 11. Failure Response is the standard response as defined in Table 4

2.3.3.2. Stop active subscriptions by id

Method: DELETE /data/subscription/id/{id}

Description: Delete active subscriptions for an id

Parameters: id

Response: Success response as defines in Table 3 and an object called "subscriptions" which contains an Array of active subscriptions. Each of these include an object as defined in Table 1 and a subscription attribute definition in a protocolextension from Table 2. The protocol extension for BLE is defined in Table 10. The protocol extension for Zigbee is defined in Table 11. Failure Response is the standard response as defined in Table 4

2.3.4. Subscriptions by topic

/data/subscription/topic/{topic}

The subscription by topic element allows an application to operate on all subscriptions of a specific topic.

Operations: - Return active subscriptions by topic: GET - Terminate all subscriptions active on a topic: DELETE

2.3.4.1. Return active subscriptions by topic

Method: GET /data/subscription/topic/{topic}

Description: Returns connection state by topic

Parameters: topic

Response: Success response as defines in Table 3 and an object called "subscriptions" which contains an Array of active subscriptions. Each of these include an object as defined in Table 1 and a subscription attribute definition in a protocolextension from Table 2. The protocol extension for BLE is defined in Table 10. The protocol extension for Zigbee is defined in Table 11. Failure Response is the standard response as defined in Table 4

2.3.4.2. Stop active subscriptions by topic

Method: DELETE /data/subscription/topic/{topic}

Description: Delete all active subscriptions for a topic

Parameters: topic

Response: Success response as defines in Table 3 and an object called "subscriptions" which contains an Array of active subscriptions. Each of these include an object as defined in Table 1 and a subscription attribute definition in a protocolextension from Table 2. The protocol extension for BLE is defined in Table 10. The protocol extension for Zigbee is defined in Table 11. Failure Response is the standard response as defined in Table 4

2.3.5. Broadcast

/data/broadcast

The broadcast element allows an application to broadcast a message to a specific device. Note that broadcasts can be heard by other devices on the same L2 network.

Operations:

*Broadcast message: POST

2.3.5.1. Broadcasting a message

Method: POST /data/broadcast

Description: Broadcasts a message to a device

Parameters: None

Request Body: an Object as defined in Table 1 along with broadcast parameters, defined below in Table 13. Defining broadcast attributes is mandatory and is done by adding an array of broadcast attributes in a protocol extension from Table 2. The protocol extension for BLE broadcasts is defined in Table 14 below.

Broadcast parameters:

Attribute	Req	Type	Example
cycle	T	enum	single
broadcastTime	F	int	30
broadcastInterval	F	int	5

Table 13: Broadcast parameters

where-

*cycle determines the repetitiveness of the broadcast, and is either single or repeat

*broadcastTime is the maximum time in seconds the broadcast should run

*broadcastInterval is the time between broadcasts in seconds

Protocol-specific extensions are supplied to identify the attributes to be broadcasted.

Attribute	Req	Type	Example
adType	T	byte	ff
adData	T	byte	4c00

Table 14: BLE Broadcast Attribute

where-

*adType is the BLE advertisement attribute type

*adData is the BLE advertisement attribute data

Response: See Table 3 for success, and Table 4 for failed responses.

2.4. Registrations

/registation

Registration elements are elements that do not directly execute operations on devices but register attributes on the gateway that support operations, such attributes are topics for data streaming and files to write files.

2.4.1. Topic registration

/registration/topic

The topic registration element allows an application to register a pub/ sub topic for the data interface. By activating a subscription on one or more device(s), the application can then publish streaming data to that topic.

Operations:

- *Register a topic: POST
- *Update a topic: PUT
- *Get configuration of one or more topics: GET
- *Delete a topic: DELETE

2.4.1.1. Registering a topic

Method: POST /registration/topic

Description: Register a pub/sub topic

Parameters: None

Request Body: A topic, including data apps that can subscribe to the topic as defined in Table 12 and protocol-specific extensions as per Table 2 that describe the attributes that will be reported on the topic. In the case of BLE, these are either BLE subscription attributes as in Table 10, device connection status, or Broadcast (advertisement) data as in Table 14. For Zigbee, these are Zigbee attributes as described in Table 11.

Response: See Table 3 with a topic name as in Table 15 below for success, and Table 4 for failed responses.

Topic name that was registered:

Attribute	Req	Type	Example
topic	T	string	"enterprise/hospital/pulse"

Table 15: Topic Name

2.4.1.2. Updating a topic

Method: PUT /registration/topic

Description: Update a pub/sub topic

Parameters: None

Request Body: A topic, including data apps that can subscribe to the topic as defined in Table 12 and protocol-specific extensions as per Table 2 that describe the attributes that will be reported on the topic. In the case of BLE, these are either BLE subscription attributes as in Table 10, device connection status, or Broadcast (advertisement) data as in Table 14. For Zigbee, these are Zigbee attributes as described in Table 11.

Response: See Table 3 with a topic name as in Table 15 below for success, and Table 4 for failed responses.

2.4.1.3. Get configuration of one or more topics

Method: GET /registration/topic

Description: Gets the configuration of one or more topics

Parameters: A topic name. Multiple topics can be added by comma-separated attributes.

Response: A success response as in Table 3 with a "topics" object containing an array of returned topics names with attribute defined in Table 15. For failed responses see Table 4.

2.4.1.4. Delete one or more topics

Method: DELETE /registration/topic

Description: Delete one or more topics

Parameters: A topic name. Multiple topics can be added by comma-separated attributes.

Response: A success response as in Table 3 with a "topics" object containing an array of returned topics names with attribute defined in Table 15. For failed responses see Table 4.

2.4.2. Topic registrations by id

/registration/topic/id/{id}

The topic registration by id element allows an application to get or delete topic registrations for a specific id.

Operations:

*Return active topics by id: GET

*Delete all topics for an id: DELETE

2.4.2.1. Return active topics by id

Method: GET /registration/topic/id/{id}

Description: Returns active topics by id

Parameters: id

Response: A success response as in Table 3 with a "topics" object containing an array of returned topics names with attribute defined in Table 15. For failed responses see Table 4.

2.4.2.2. Delete active topics by id

Method: DELETE /registration/topic/id/{id}

Description: Deletes active topics by id, will delete all topics that are associated with a specific object id.

Parameters: id

Response: A success response as in Table 3 with a "topics" object containing an array of returned topics names with attribute defined in Table 15. For failed responses see Table 4.

2.4.3. Topic registrations by data app

/registration/topic/data-app/{data-app}

The topic registration by data-app element allows an application to get or delete topic registrations for which a specific data-application is registered.

Operations:

*Return active topics by data-app: GET

*Delete all topics for an data-app: DELETE

2.4.3.1. Return active topics by data-app

Method: GET /registration/topic/data-app/{data-app}

Description: Returns active topics by data-app

Parameters: data-app

Response: A success response as in Table 3 with a "topics" object containing an array of returned topics names with attribute defined in Table 15. For failed responses see Table 4.

2.4.3.2. Delete active topics by data-app

Method: DELETE /registration/topic/data-app/{data-app}

Description: Deletes active topics by data-app, will delete all topics the specified data app is registered for.

Parameters: data-app

Response: A success response as in Table 3 with a "topics" object containing an array of returned topics names with attribute defined in Table 15. For failed responses see Table 4.

2.4.4. Topic registrations by topic name

/registration/topic/{topic}

The topic registration by topic element allows an application to get or delete a topic registration by topic name.

Operations:

- *Return active topics by topic name: GET
- *Delete all topics for a topic name: DELETE

2.4.4.1. Return active topics by topic name

Method: GET /registration/topic/{topic}

Description: Returns active topics by topic name

Parameters: topic

Response: A success response as in Table 3 with a topic name as defined in Table 15. For failed responses see Table 4.

2.4.4.2. Delete active topics by topic name

Method: DELETE /registration/topic/{topic}

Description: Deletes active topics by topic name

Parameters: topic

Response: A success response as in Table 3 with a topic name as defined in Table 15. For failed responses see Table 4.

2.4.5. File registration

/registration/file

The file registration element allows an application to register a file. a file can be used in operations to devices (such as an attribute)

Operations:

*Register a file: POST

*Update a file: PUT

*Check if a file exists: GET

*Delete a file: DELETE

2.4.5.1. Registering a file

Method: POST /registration/file

Description: Register a file

Parameters: None

Request Body: a file or URL point to a file, as described in Table 16 below.

File definition:

Attribute	Req	Type	Example
filename	T	string	"firmware.dat"
file	F	binary	file
bindings	F	string	"https://domain.com/firmware.dat"

Table 16: File

Response: See Table 3 for success, and Table 4 for failed responses.

2.4.5.2. Updating a file

Method: PUT /registration/file

Description: Update a file

Parameters: None

Request Body: a file or URL point to a file, as described in Table 16 below.

Response: See Table 3 for success, and Table 4 for failed responses.

2.4.5.3. check presence of a registered file

Method: GET /registration/file

Description: Check the presence of a specific file or get all files if no file name os present in parameters

Parameters: filename

Response: Success as in Table 3 including a "filesnames" object with an array of file names as shown in Table 17 or a Failure Response as described in Table 4.

A filenames object with an Array of file names:

Attribute	Req	Type	Example
filename	T	string	"firmware.dat"

Table 17: File name

2.4.5.4. delete a registered file

Method: DELETE /registration/file

Description: Delete a registered file

Parameters: filename

Response: See Table 3 for success, and Table 4 for failed responses.

2.4.6. file registrations by file name

/registration/file/{filename}

The file registration by file name element allows an application to get or delete file registrations by file name.

Operations:

*Return file registrations by file name: GET

*Delete file registrations by file name: DELETE

2.4.6.1. Return file registrations by file name

Method: GET /registration/file/{filename}

Description: Checks the presence of a file and returns its name

Parameters: filename

Response: Success as in Table 3 including a file name as shown in Table 17 or a Failure Response as described in Table 4.

2.4.6.2. Delete file registrations by file name

Method: DELETE /registration/file/{filename}

Description: Delete file registrations by file name

Parameters: filename

Response: See Table 3 for success, and Table 4 for failed responses.

3. NIPC Extensibility

NIPC is extensible in two ways:

- Protocol extensions: Protocol extensions can extend NIPC with support for new non-IP protocols

- interface extensions: Interface extensions allow extensions that leverage compound statements of basic elements to simplify common operations for applications.

3.1. Protocol extensions

As described in Figure 3 the NIPC interface supports protocol specific extensions that allow bi-directional communication of attributes that are specific to the protocol supported by the device. This allows for extensions to the schema to integrate new non-ip communications protocols, without the need to update the base schema.

```
ID
 - device/group attributes
    |
    |> BLE
    |     - BLE attributes
    |
    |> Zigbee
    |     - Zigbee attributes
    |
    |> Protocol extension
    |     - Protocol extension attributes
```

Figure 5: Extended Schema

As shown in Figure 5, a protocol extension can be added by adding a new technology specific extension to the schema.

This is performed by adding the new protocol to the technology enum in the base objext definition Table 1

Furthermore, the protocol objects need to be extended with the new protocol as well. Protocol objects will be extended as follows:

Attribute	Req	Type	Example
ble	T	object	an object with BLE-specific attributes
zigbee	T	object	an object with Zigbee-specific attributes
newProtocol	T	object	an object with newProtocol-specific attr

Table 18: Adding Protocol extensions

In the new protocol object, protocol specific attributes can be added.

3.2. Interface extensions

/extensions

The interface extension elements are freely extendible interfaces. These elements leverage the basic NIPC defined elements and combine them in compound statements in order to streamline application operation against devices, make operations more expediant and convenient in one API call. In principle they do not add any basic functionality. In the OpenAPI model Figure 6 below, we have defined a few example extensions, and we will describe them here at a high level to provide some context on other possible extensions.

3.2.1. Write file

/extension/write/file

This extension make use of multiple write operations (attribute post) to write an entire file to an attribute. The interface allows the application to define the chunk size.

Operations:

*Write file: POST

3.2.2. Read conditional file

/extension/read/conditional

This extension performs a read operation sequentially for a defined amount of time until a specified value is read.

Operations:

*Read Conditional: POST

3.2.3. Bulk

/extension/bulk

This extension allows you to create a compound operation made out of multiple connection and data operations that are to be executed sequentially until they all succeed or there is a failure. Supported operations are:

*/extension/connection/create

*/extension/connection/delete

*/extension/attribute/read

*/extension/attribute/write

*/extension/attribute/write/file

*/extension/attribute/write/blob

*/extension/attribute/read/conditional

Operations:

*Bulk: POST

4. Publish/Subscribe Interface

The publish/subscribe interface, or data streaming interface, is an MQTT publishing interface. Pub/sub topics can be created and managed by means of the /register/topic NIPC element.

In this memo we propose the data format to be protocol buffers, as fully described in the Figure 7 protobuf definition.

5. Examples

This section contains a few examples on how applications can leverage NIPC operations to communicate with BLE and Zigbee devices.

5.1. BLE Advertisement

In this example, we will onboard a device, and setup an advertisement subscription topic for that device.

The sequence of operations for this are:

* Onboard a device using the SCIM Interface (out of scope of this memo)

* Register a topic with the device id to subscribe to advertisements POST /register/topic

* Subscribe to the topic from the data receiver app MQTT subscribe topic

5.2. BLE Attribute Read/Write

In this example, we will connect to a BLE device (BLE device does not require binding) and read and write from an attribute

The sequence of operations for this are:

* Onboard a device using the SCIM Interface (out of scope of this memo)

* Connect to the BLE device POST /connectivity/connection

* Read an attribute from the BLE device GET /data/attribute

* Write to an attribute on the BLE device POST /data/attribute

* Disconnect from the BLE device DELETE /connectivity/connection

5.3. Zigbee Attribute Read/Write

In this example, we will bind a zigbee device to a Zigbee mesh and read and write from an attribute

The sequence of operations for this are:

* Onboard a device using the SCIM Interface (out of scope of this memo)

*Bind the Zigbee device POST /connectivity/binding

*Read an attribute from the Zigbee device GET /data/attribute

*Write to an attribute on the Zigbee device POST /data/attribute

*Disconnect from the Zigbee device DELETE /connectivity/connection

6. Security Considerations

TBD.

7. IANA Considerations

TBD.

8. Normative References

[BLE53] Bluetooth SIG, "Bluetooth Core Specification, Version 5.3", 2021.

[I-D.shahzad-scim-device-model] Shahzad, M., Hassan, H., and E. Lear, "Device Schema Extensions to the SCIM model", Work in Progress, Internet-Draft, draft-shahzad-scim-device-model-05, 2 June 2023, <https://datatracker.ietf.org/doc/html/draft-shahzad-scim-device-model-05>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

[Zigbee22] Connectivity Standards Alliance, "zigbee Specification, Version 22 1.0", 2017.

Appendix A. OpenAPI definition

The following non-normative model is provide for convenience of the implementor.

<CODE BEGINS> file "openapi.yml"

```
openapi: 3.0.3
info:
  title: Non IP Device Control (NIPC) API
  description: |-
    There has been a large influx of non-IP devices supporting
    processes in manufacturing, healthcare, hospitality, retail, the
    home, and the office.  At the same time, wireless access points
    have been deployed nearly everywhere, many of which have radios
    that can transmit and receive different frame types, such as BLE,
    Zigbee. To integrate multiple of these use cases leveraging a
    single wireless infrastructure and avoid the need for parallel
    infrastructure, a Non IP device gateway function is necessary.
    The gateway provides the following functions:
       - authentication and authorization of application clients that
         will communicate with devices
       - APIs that onboard a device on the network (out of scope for
         this specification, but covered in SCIM for devices)
       - APIs that allow an app to set up a connection with a device
       - APIs that allow an app to exchange data with a device
       - APIs that allow a device to create registrations in the
         network for a device
    These collection of these APIs, in combination with the
    onboarding API (SCIM for devices) will allow an application to
    perform a complete set of operations on Non-IP devices.
  termsOfService: http://swagger.io/terms/
  contact:
    email: bbrinckm@cisco.com
  license:
    name: TBD
    url: TBD
  version: 0.5.3
externalDocs:
  description: NIPC IETF draft
  url: TBD
servers:
  - url: https://{gw_host}/nipc
    variables:
      gw_host:
        default: localhost
        description: Gateway Host
tags:
  - name: connectivity
    description: APIs that allow apps to manage device connections
  - name: data
    description: |-
      APIs that allow apps to exchange data with non-IP devices
  - name: registrations
```

```
    description: |-
      APIs that allow apps to make registrations in the network for
      devices.
  - name: extensions
    description: |-
      APIs that simplify application interaciton by implementing one
      or more basic API's into a single API call.

paths:
### Connectivity
  /connectivity/binding:
    post:
      tags:
        - connectivity
      summary: Create a binding for a device id
      description: Create a binding for a device
      operationId: CreateBinding
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Object'
          application/xml:
            schema:
              $ref: '#/components/schemas/Object'
          application/x-www-form-urlencoded:
            schema:
              $ref: '#/components/schemas/Object'
        required: true
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/BindingResponse'
            application/xml:
              schema:
                $ref: '#/components/schemas/BindingResponse'
        '400':
          description: Bad request
        '401':
          description: Unauthorized
        '405':
          description: Invalid request
        '500':
          description: Server-side failure
          content:
            application/json:
```

```
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

get:
  tags:
    - connectivity
  summary: Get bindings for a device
  description: |-
    Get all bindings control app made (no parameter) or binding
    by object ID, Multiple ids can be provided with comma
    separated strings, or a group id can be provided
  operationId: GetBindings
  parameters:
    - name: id
      in: query
      description: device ids that need to be filtered
      required: false
      explode: true
      schema:
        type: string
        format: uuid
        example: 12345678-1234-5678-1234-56789abcdef4
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/MultiBindingsResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/MultiBindingsResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
```

```
              $ref: '#/components/schemas/FailureResponse'
  delete:
    tags:
      - connectivity
    summary: |-
      Delete bindings for a device or group of devices
    description: |-
      Delete all bindings control app made or bindings by object
      ID, Multiple ids can be provided with comma separated
      strings, or a group id can be provided
    operationId: DeleteBindings
    parameters:
      - name: id
        in: query
        description: device or group ids that need to be filtered
        required: false
        explode: true
        schema:
          type: string
          format: uuid
          example: 12345678-1234-5678-1234-56789abcdef4
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/connectivity/binding/id/{id}:
  post:
```

```
  tags:
    - connectivity
  summary: |-
    Create a binding for a device id (device technology needs to
    support binding)
  description: |-
    Create a binding for a device id, will fail if device has
    multiple technologies defined
  operationId: CreateBindingbyID
  parameters:
    - name: id
      in: path
      description: device or group ids that need to be filtered
      required: true
      schema:
        type: string
        format: uuid
        example: 12345678-1234-5678-1234-56789abcdef4
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/BindingResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/BindingResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

get:
  tags:
    - connectivity
  summary: Get binding by id for a device
  description: |-
```

```
    Get binding by id for a device, success when binding found,
    failure when no binding
  operationId: GetBindingbyId
  parameters:
    - name: id
      in: path
      description: device or group ids that need to be filtered
      required: true
      schema:
        type: string
        format: uuid
        example: 12345678-1234-5678-1234-56789abcdef4
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/BindingResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/BindingResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

delete:
  tags:
    - connectivity
  summary: Delete binding by id for a device
  description: Delete binding by id for a device
  operationId: DeleteBindingbyID
  parameters:
    - name: id
      in: path
      description: device or group ids that need to be filtered
      required: true
```

```
        schema:
          type: string
          format: uuid
          example: 12345678-1234-5678-1234-56789abcdef4
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/connectivity/connection:
  post:
    tags:
      - connectivity
    summary: |-
      Connect a device to the network, optionally with service
      discovery
    description: |-
      Connect a device to the network, optionally with service
      discovery
    operationId: connConnect
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Connection'
        application/xml:
          schema:
            $ref: '#/components/schemas/Connection'
```

```
      application/x-www-form-urlencoded:
        schema:
          $ref: '#/components/schemas/Connection'
    required: true
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ServiceResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/ServiceResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

get:
  tags:
    - connectivity
  summary: |-
    Get connection state devices for a device or group of
    devices
  description: |-
    Get all connection status for connections made by control ap
    or connection status by object ID, multiple ids can be
    provided with comma separated strings, or a group id can be
    provided
  operationId: GetConnections
  parameters:
    - name: id
      in: query
      description: device or group ids that need to be filtered
      required: false
      explode: true
      schema:
```

```
        type: string
        format: uuid
        example: 12345678-1234-5678-1234-56789abcdef4
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/MultiConnectionsResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/MultiConnectionsResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

  delete:
    tags:
      - connectivity
    summary: Disconnect a device or group of devices
    description: |-
      Disconnect a device or device group by object ID, Multiple
      ids can be provided with comma separated strings, or a
      group id can be provided
    operationId: DeleteConnections
    parameters:
      - name: id
        in: query
        description: device or group ids that need to be filtered
        required: false
        explode: true
        schema:
          type: string
          format: uuid
          example: 12345678-1234-5678-1234-56789abcdef4
    responses:
```

```
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/connectivity/connection/id/{id}:
  post:
    tags:
      - connectivity
    summary: |-
      Connect a device by device id (device technology needs to
      support connection)
    description: |-
      Connect a device by device id, Serivce discovery not
      supported, will fail if device has multiple technologies
      defined.
    operationId: CreateConnectionbyID
    parameters:
      - name: id
        in: path
        description: device or group ids that need to be filtered
        required: true
        schema:
          type: string
          format: uuid
          example: 12345678-1234-5678-1234-56789abcdef4
    responses:
      '200':
        description: Success
```

```
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

get:
  tags:
    - connectivity
  summary: Get connection by id for a device
  description: |-
    Get connection by id for a device, success when device
    connected, failure when device not connected
  operationId: GetConnectionbyId
  parameters:
    - name: id
      in: path
      description: device or group ids that need to be filtered
      required: true
      schema:
        type: string
        format: uuid
        example: 12345678-1234-5678-1234-56789abcdef4
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
        application/xml:
          schema:
```

```
            $ref: '#/components/schemas/SuccessResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

delete:
  tags:
    - connectivity
  summary: Delete connection by id for a device
  description: Disconnect a device by id
  operationId: DeleteConnectionbyID
  parameters:
    - name: id
      in: path
      description: device or group ids that need to be filtered
      required: true
      schema:
        type: string
        format: uuid
        example: 12345678-1234-5678-1234-56789abcdef4
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
```

```
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/connectivity/services:
  get:
    tags:
      - connectivity
    summary: Discover services on a device
    description: Discover services on a device
    operationId: ServiceDiscovey
    parameters:
    - name: Service
      in: query
      description: Services to discover
      required: true
      schema:
        $ref: '#/components/schemas/Service'
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ServiceResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/ServiceResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/connectivity/services/id/{id}:
  get:
    tags:
```

```
        - connectivity
      summary: Get services by id for a device
      description: |-
        Get services by id for a connected device, success when
        device connected, failure when device not connected
      operationId: GetServicesbyId
      parameters:
        - name: id
          in: path
          description: device or group ids that need to be filtered
          required: true
          schema:
            type: string
            format: uuid
            example: 12345678-1234-5678-1234-56789abcdef4
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ServiceResponse'
            application/xml:
              schema:
                $ref: '#/components/schemas/ServiceResponse'
        '400':
          description: Bad request
        '401':
          description: Unauthorized
        '405':
          description: Invalid request
        '500':
          description: Server-side failure
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/FailureResponse'
            application/xml:
              schema:
                $ref: '#/components/schemas/FailureResponse'

### Data
  /data/attribute:
    post:
      tags:
        - data
      summary: Write a value to an attribute on a device
      description: Write a value to an attribute on a device
      operationId: dataWrite
```

```
  requestBody:
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/AttributeValue'
      application/xml:
        schema:
          $ref: '#/components/schemas/AttributeValue'
      application/x-www-form-urlencoded:
        schema:
          $ref: '#/components/schemas/AttributeValue'
    required: true
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AttributeValueResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/AttributeValueResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

put:
  tags:
    - data
  summary: Update a value of an attribute on a device
  description: Update a value of an attribute on a device
  operationId: dataUpdate
  requestBody:
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/AttributeValue'
```

```
      application/xml:
        schema:
          $ref: '#/components/schemas/AttributeValue'
      application/x-www-form-urlencoded:
        schema:
          $ref: '#/components/schemas/AttributeValue'
    required: true
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AttributeValueResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/AttributeValueResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

delete:
  tags:
    - data
  summary: Delete a value from an attribute on a device
  description: Delete a value to an attribute on a device
  operationId: dataDelete
  parameters:
  - name: attribute
    in: query
    description: attributes of a given device
    required: true
    schema:
      $ref: '#/components/schemas/Attribute'
  responses:
    '200':
      description: Success
```

```
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AttributeValueResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/AttributeValueResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

get:
  tags:
    - data
  summary: Read a value from an attribute on a device
  description: Read a value to an attribute on a device
  operationId: dataRead
  parameters:
  - name: attribute
    in: query
    description: attributes of a given device
    required: true
    schema:
      $ref: '#/components/schemas/Attribute'
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AttributeValueResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/AttributeValueResponse'
    '400':
      description: Bad request
    '401':
```

```
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/data/subscription:
  post:
    tags:
      - data
    summary: |-
       Subscribe to streaming data from an attribute on a device
    description: |-
      Subscribe to streaming data from an attribute on a device
    operationId: dataSubscribe
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Subscription'
        application/xml:
          schema:
            $ref: '#/components/schemas/Subscription'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/Subscription'
      required: true
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
```

```
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

put:
  tags:
    - data
  summary: |-
    update streaming data subscription from an attribute on a
    device
  description: |-
    update streaming data subscription from an attribute on a
    device
  operationId: dataUpdateSubscribe
  requestBody:
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Subscription'
      application/xml:
        schema:
          $ref: '#/components/schemas/Subscription'
      application/x-www-form-urlencoded:
        schema:
          $ref: '#/components/schemas/Subscription'
    required: true
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
```

```
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

delete:
  tags:
    - data
  summary: |-
    Unsubscribe to streaming data from an attribute on a device
  description: |-
    Unsubscribe to streaming data from an attribute on a device
  operationId: dataUnsubscribe
  parameters:
  - name: subscription
    in: query
    description: subscription on a device
    required: true
    schema:
      $ref: '#/components/schemas/Subscription'
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
```

```
              $ref: '#/components/schemas/FailureResponse'

  get:
    tags:
      - data
    summary: Get the status of a subscription on a device
    description: Get the status of a subscription on a device
    operationId: dataGetSubscription
    parameters:
    - name: subscription
      in: query
      description: subscription on a device
      required: true
      schema:
        $ref: '#/components/schemas/Subscription'
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/data/subscription/topic/{topic}:
  delete:
    tags:
      - data
    summary: delete all active subscriptions by topic
    description: delete all active subscriptions by topic
    operationId: deleteSubscriptionbyTopic
    parameters:
```

```
    - name: topic
      in: path
      description: topic that needs to be filtered
      required: true
      schema:
        type: string
        example: "enterprise/hospital/pulse_oximeter"
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref:
               '#/components/schemas/MultiSubscriptionResponse'
        application/xml:
          schema:
            $ref:
               '#/components/schemas/MultiSubscriptionResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

get:
  tags:
    - data
  summary: get all active subscriptions by topic
  description: get all active subscriptions by topic
  operationId: getSubsciptionsbyTopic
  parameters:
    - name: topic
      in: path
      description: topic that needs to be filtered
      required: true
      schema:
        type: string
        example: "enterprise/hospital/pulse_oximeter"
```

```
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref:
                 '#/components/schemas/MultiSubscriptionResponse'
          application/xml:
            schema:
              $ref:
                 '#/components/schemas/MultiSubscriptionResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref:
                 '#/components/schemas/MultiSubscriptionResponse'
          application/xml:
            schema:
              $ref:
                 '#/components/schemas/MultiSubscriptionResponse'

/data/subscription/id/{id}:
  delete:
    tags:
      - data
    summary: delete all subscriptions by id
    description: delete all subscriptions by id
    operationId: deleteSubscriptionbyID
    parameters:
      - name: id
        in: path
        description: object id that needs to be filtered
        required: true
        schema:
          type: string
          example: 12345678-1234-5678-1234-56789abcdef4
    responses:
      '200':
        description: Success
        content:
```

```
        application/json:
          schema:
            $ref:
               '#/components/schemas/MultiSubscriptionResponse'
        application/xml:
          schema:
            $ref:
               '#/components/schemas/MultiSubscriptionResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

get:
  tags:
    - data
  summary: get all subscriptions by object id
  description: get all subscriptions by object id
  operationId: getSubscriptionbyID
  parameters:
    - name: id
      in: path
      description: object id that needs to be filtered
      required: true
      schema:
        type: string
        example: 12345678-1234-5678-1234-56789abcdef4
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref:
               '#/components/schemas/MultiSubscriptionResponse'
        application/xml:
          schema:
            $ref:
```

```
                '#/components/schemas/MultiSubscriptionResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/data/broadcast:
  post:
    tags:
      - data
    summary: Broadcast to a device
    description: Broadcast to a device
    operationId: dataBroadcast
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Broadcast'
        application/xml:
          schema:
            $ref: '#/components/schemas/Broadcast'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/Broadcast'
      required: true
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
      '400':
        description: Bad request
      '401':
```

```
          description: Unauthorized
        '405':
          description: Invalid request
        '500':
          description: Server-side failure
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/FailureResponse'
            application/xml:
              schema:
                $ref: '#/components/schemas/FailureResponse'

### Registrations
  /registration/topic:
    post:
      tags:
        - registrations
      summary: Register a publish/subscribe topic
      description: Register a publish/subscribe topic
      operationId: registerTopic
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Topic'
          application/xml:
            schema:
              $ref: '#/components/schemas/Topic'
          application/x-www-form-urlencoded:
            schema:
              $ref: '#/components/schemas/Topic'
        required: true
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/TopicResponse'
            application/xml:
              schema:
                $ref: '#/components/schemas/TopicResponse'
        '400':
          description: Bad request
        '401':
          description: Unauthorized
        '405':
          description: Invalid request
```

```
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

put:
  tags:
    - registrations
  summary: Update a publish/subscribe topic
  description: Update a publish/subscribe topic
  operationId: UpdateTopic
  requestBody:
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Topic'
      application/xml:
        schema:
          $ref: '#/components/schemas/Topic'
      application/x-www-form-urlencoded:
        schema:
          $ref: '#/components/schemas/Topic'
    required: true
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/TopicResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/TopicResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
```

```
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

delete:
  tags:
    - registrations
  summary: unregister a publish/subscribe topic
  description: |-
    unregister a publish/subscribe topic, Multiple topics can
    be provided with comma separate strings, or a group id can
    be provided.
  operationId: unregisterTopic
  parameters:
    - name: topic
      in: query
      description: topic that need to be filtered
      required: false
      explode: true
      schema:
        type: string
        example: "enterprise/hospital/pulse_oximeter"
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/MultiTopicsResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/MultiTopicsResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'
```

```
  get:
    tags:
      - registrations
    summary: get one or multiple publish/subscribe topic
    description: |-
      unregister a publish/subscribe topic, Multiple topics can be
      provided with comma separate strings, or a group id can be
      provided
    operationId: getTopic
    parameters:
      - name: topic
        in: query
        description: topic that need to be filtered
        required: false
        explode: true
        schema:
          type: string
          example: "enterprise/hospital/pulse_oximeter"
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/MultiTopicsResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/MultiTopicsResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/registration/topic/{topic}:
  delete:
    tags:
      - registrations
    summary: delete a publish/subscribe topic by name
```

```
    description: unregister a publish/subscribe topic by Name
    operationId: deleteTopicbyName
    parameters:
      - name: topic
        in: path
        description: topic that needs to be filtered
        required: true
        schema:
          type: string
          example: "enterprise/hospital/pulse_oximeter"
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/TopicResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/TopicResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

  get:
    tags:
      - registrations
    summary: get a publish/subscribe topic by name
    description: get a publish/subscribe topic by name
    operationId: getTopicbyName
    parameters:
      - name: topic
        in: path
        description: topic that needs to be filtered
        required: true
        schema:
          type: string
```

```
          example: "enterprise/hospital/pulse_oximeter"
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/TopicResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/TopicResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/registration/topic/data-app/{data-app}:
  delete:
    tags:
      - registrations
    summary: delete all publish/subscribe topics by data-app
    description: |-
      unregister all publish/subscribe topics by data-app
    operationId: deleteTopicbyDataApp
    parameters:
      - name: data-app
        in: path
        description: data app that needs to be filtered
        required: true
        schema:
          type: string
          example: https://data-app-1
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
```

```
            $ref: '#/components/schemas/MultiTopicsResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/MultiTopicsResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

get:
  tags:
    - registrations
  summary: get all publish/subscribe topics by data-app
  description: get all publish/subscribe topics by data-app
  operationId: getTopicbyDataApp
  parameters:
    - name: data-app
      in: path
      description: data app that needs to be filtered
      required: true
      schema:
        type: string
        example: https://data-app-1
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/MultiTopicsResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/MultiTopicsResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
```

```
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/registration/topic/id/{id}:
  delete:
    tags:
      - registrations
    summary: delete all publish/subscribe topics by object id
    description: unregister all publish/subscribe topics by id
    operationId: deleteTopicbyID
    parameters:
      - name: id
        in: path
        description: object id that needs to be filtered
        required: true
        schema:
          type: string
          example: 12345678-1234-5678-1234-56789abcdef4
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/MultiTopicsResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/MultiTopicsResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
```

```
            schema:
              $ref: '#/components/schemas/FailureResponse'

  get:
    tags:
      - registrations
    summary: get all publish/subscribe topics by object id
    description: get all publish/subscribe topics by object id
    operationId: getTopicbyID
    parameters:
      - name: id
        in: path
        description: object id that needs to be filtered
        required: true
        schema:
          type: string
          example: 12345678-1234-5678-1234-56789abcdef4
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/MultiTopicsResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/MultiTopicsResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/registration/file:
  post:
    tags:
      - registrations
    summary: Register and upload a file for later use
    description: Register and upload a file for later use
```

```
    operationId: registerFile
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/File'
        application/xml:
          schema:
            $ref: '#/components/schemas/File'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/File'
      required: true
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

  put:
    tags:
      - registrations
    summary: Update an existing file registration
    description: Update an existing file registration
    operationId: UpdateFile
    requestBody:
      content:
        application/json:
          schema:
```

```
          $ref: '#/components/schemas/File'
      application/xml:
        schema:
          $ref: '#/components/schemas/File'
      application/x-www-form-urlencoded:
        schema:
          $ref: '#/components/schemas/File'
    required: true
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

delete:
  tags:
    - registrations
  summary: Delete a file
  description: Delete a file
  operationId: DeleteFile
  parameters:
    - name: filename
      in: query
      description: file that needs to be filtered
      required: false
      explode: true
      schema:
        type: string
        example: "firmware.dat"
```

```
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/SuccessResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

get:
  tags:
    - registrations
  summary: get a file
  description: |-
    get a file by name of get all files if no names
    supplied.
  operationId: getFile
  parameters:
    - name: filename
      in: query
      description: file that needs to be filtered
      required: false
      explode: true
      schema:
        type: string
        example: "firmware.dat"
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
```

```
              $ref: '#/components/schemas/MultiFileResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/MultiFileResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/registration/file/{filename}:
  delete:
    tags:
      - registrations
    summary: delete a file by name
    description: delete a file by name
    operationId: deleteFilebyName
    parameters:
      - name: filename
        in: path
        description: file that needs to be filtered
        required: true
        schema:
          type: string
          example: "firmware.dat"
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FileResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FileResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
```

```
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FailureResponse'

get:
  tags:
    - registrations
  summary: get a file by name
  description: get a file by name
  operationId: getFilebyName
  parameters:
    - name: filename
      in: path
      description: file that needs to be filtered
      required: true
      schema:
        type: string
        example: 12345678-1234-5678-1234-56789abcdef4
  responses:
    '200':
      description: Success
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FileResponse'
        application/xml:
          schema:
            $ref: '#/components/schemas/FileResponse'
    '400':
      description: Bad request
    '401':
      description: Unauthorized
    '405':
      description: Invalid request
    '500':
      description: Server-side failure
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/FailureResponse'
        application/xml:
```

```
              schema:
                $ref: '#/components/schemas/FailureResponse'

### Extensions
  /extension/connection/create:
    post:
      tags:
        - extensions
      summary: |-
        Connect a device to the network, optionally with service
        discovery
      description: |-
        Connect a device to the network, optionally with service
        discovery
      operationId: ExtConnect
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Connection'
          application/xml:
            schema:
              $ref: '#/components/schemas/Connection'
          application/x-www-form-urlencoded:
            schema:
              $ref: '#/components/schemas/Connection'
        required: true
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ServiceResponse'
            application/xml:
              schema:
                $ref: '#/components/schemas/ServiceResponse'
        '400':
          description: Bad request
        '401':
          description: Unauthorized
        '405':
          description: Invalid request
        '500':
          description: Server-side failure
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/FailureResponse'
```

```
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/extension/connection/delete:
  post:
    tags:
      - extensions
    summary: |-
      Connect a device to the network, optionally with service
      discovery
    description: |-
      Connect a device to the network, optionally with service
      discovery
    operationId: ExtDisconnect
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Object'
        application/xml:
          schema:
            $ref: '#/components/schemas/Object'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/Object'
      required: true
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
```

```
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/extension/attribute/write:
  post:
    tags:
      - extensions
    summary: Write a value to an attribute on a device
    description: Write a value to an attribute on a device
    operationId: dataAttrWrite
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AttributeValue'
        application/xml:
          schema:
            $ref: '#/components/schemas/AttributeValue'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/AttributeValue'
      required: true
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/AttributeValueResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/AttributeValueResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'
```

```
/extension/attribute/write/file:
  post:
    tags:
      - extensions
    summary: Write a file to an attribute across multiple writes
    description: |-
      Write a file to an attribute across multiple writes
    operationId: dataWriteFile
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AttributeFile'
        application/xml:
          schema:
            $ref: '#/components/schemas/AttributeFile'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/AttributeFile'
      required: true
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/extension/attribute/write/blob:
  post:
    tags:
```

```
      - extensions
    summary: |-
      Write a binary blob to an attribute across multiple writes
    description: |-
      Write a binary blob to an attribute across multiple writes
    operationId: dataWriteBlob
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AttributeBlob'
        application/xml:
          schema:
            $ref: '#/components/schemas/AttributeBlob'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/AttributeBlob'
      required: true
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/SuccessResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/extension/attribute/read:
  post:
    tags:
      - extensions
    summary: Read an attribute on a device
```

```
    description: Write a value to an attribute on a device
    operationId: dataAttrRead
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Attribute'
        application/xml:
          schema:
            $ref: '#/components/schemas/Attribute'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/Attribute'
      required: true
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/AttributeValueResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/AttributeValueResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/extension/attribute/read/conditional:
  post:
    tags:
      - extensions
    summary: |-
      Read a value from attribute on a device until it matches a
      specific value.
    description: |-
      Read a value from attribute on a device until it matches a
```

```
      specific value.
    operationId: dataReadCond
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/AttributeConditional'
        application/xml:
          schema:
            $ref: '#/components/schemas/AttributeConditional'
        application/x-www-form-urlencoded:
          schema:
            $ref: '#/components/schemas/AttributeConditional'
      required: true
    responses:
      '200':
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/AttributeValueResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/AttributeValueResponse'
      '400':
        description: Bad request
      '401':
        description: Unauthorized
      '405':
        description: Invalid request
      '500':
        description: Server-side failure
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/FailureResponse'
          application/xml:
            schema:
              $ref: '#/components/schemas/FailureResponse'

/extension/bulk:
  post:
    tags:
      - extensions
    summary: Compound operations on a device
    description: Compound operations on a device
    operationId: Bulk
    requestBody:
      content:
```

```
          application/json:
            schema:
              $ref: '#/components/schemas/Bulk'
          application/xml:
            schema:
              $ref: '#/components/schemas/Bulk'
          application/x-www-form-urlencoded:
            schema:
              $ref: '#/components/schemas/Bulk'
        required: true
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/BulkResponse'
            application/xml:
              schema:
                $ref: '#/components/schemas/BulkResponse'
        '400':
          description: Bad request
        '401':
          description: Unauthorized
        '405':
          description: Invalid request
        '500':
          description: Server-side failure
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/FailureResponse'
            application/xml:
              schema:
                $ref: '#/components/schemas/FailureResponse'

components:
  schemas:
# BLE objects
##  An array for BLE services
    BLEServiceslist:
      required:
        - services
      type: object
      properties:
        services:
          type: array
          xml:
            name: services
```

```
            wrapped: true
          items:
            $ref: '#/components/schemas/BLEService'
      xml:
        name: BLEServiceslist

## A BLE service with its characteristics
    BLEService:
      required:
        - serviceID
        - characteristics
      type: object
      properties:
        serviceID:
          type: string
          format: uuid
          example: 12345678-1234-5678-1234-56789abcdef4
        characteristics:
          type: array
          xml:
            name: characteristics
            wrapped: true
          items:
            $ref: '#/components/schemas/BLECharacteristic'
      xml:
        name: BLEService

## A BLE characteristics with its descriptors
    BLECharacteristic:
      required:
        - characteristicID
        - flags
        - descriptors
      type: object
      properties:
        characteristicID:
          type: string
          format: uuid
          example: 12345678-1234-5678-1234-56789abcdef4
        flags:
          type: array
          example:
          - read
          - write
          items:
            type: string
            enum:
              - read
              - write
```

```
              - notify
        descriptors:
          type: array
          xml:
            name: descriptors
            wrapped: true
          items:
            $ref: '#/components/schemas/BLEDescriptor'
      xml:
        name: BLECharacteristic

## A BLE descriptor
    BLEDescriptor:
      required:
        - descriptorID
      type: object
      properties:
        descriptorID:
          type: string
          format: uuid
          example: 12345678-1234-5678-1234-56789abcdef4
      xml:
        name: BLEDescriptor

## BLE service ID only
    BLEServiceID:
      type: object
      properties:
        serviceID:
          type: string
          format: uuid
          example: 12345678-1234-5678-1234-56789abcdef4
      xml:
        name: BLEServiceID

## Attributes that define a BLE attribute
    BLEAttributes:
      required:
        - ble
      type: object
      properties:
        ble:
          required:
            - serviceID
            - characteristicID
          type: object
          properties:
            serviceID:
              type: string
```

```
              format: uuid
              example: 12345678-1234-5678-1234-56789abcdef4
            characteristicID:
              type: string
              format: uuid
              example: 12345678-1234-5678-1234-56789abcdef4
            long:
              type: boolean
              example: false
      xml:
        name: BLEAttributes

## Defines different types of BLE topics
    BLETopic:
      required:
        - ble
      type: object
      properties:
        ble:
          oneOf:
            - $ref: '#/components/schemas/BLESubTopic'
            - $ref: '#/components/schemas/BLEConnTopic'
            - $ref: '#/components/schemas/BLEAdvTopic'
      xml:
        name: BLETopic

## BLE Gatt Topic definition
    BLESubTopic:
      required:
        - type
        - serviceID
        - characteristicID
      type: object
      properties:
        type:
          type: string
          example: gatt
          enum:
            - gatt
        serviceID:
          type: string
          example: 12345678-1234-5678-1234-56789abcdef0
        characteristicID:
          type: string
          example: 12345678-1234-5678-1234-56789abcdef1
      xml:
        name: BLESubTopic

## BLE Connection event Topic definition
```

```
    BLEConnTopic:
      required:
        - type
      type: object
      properties:
        type:
          type: string
          example: connection_events
          enum:
            - connection_events
      xml:
        name: BLEConnTopic

## BLE Advertisement Topic definition
    BLEAdvTopic:
      required:
        - type
      type: object
      properties:
        type:
          type: string
          example: advertisements
          enum:
            - advertisements
        filterType:
          type: string
          example: deny
          enum:
            - deny
            - allow
        filters:
          type: array
          xml:
            name: filters
            wrapped: true
          items:
            $ref: '#/components/schemas/BLEAdvertisement'
      xml:
        name: BLEAdvTopic

## BLE Advertisement attributes
    BLEAdvertisement:
      type: object
      properties:
        adTtype:
          type: string
          format: byte
          example: ff
        adData:
```

```
          type: string
          format: byte
          example: 4c00*
      xml:
        name: BLEAdvertisement

## Attributes that define a BLE broadcast
    BLEBroadcast:
      required:
        - ble
      type: object
      properties:
        ble:
          required:
            - advertisement
          type: array
          xml:
            name: services
            wrapped: true
          items:
            $ref: '#/components/schemas/BLEAdvertisement'
      xml:
        name: BLEBroadcast

# Zigbee objects
## An array for Zigbee Endpoints
    ZigbeeEndpointlist:
      required:
        - endpoints
      type: object
      properties:
        endpoints:
          type: array
          xml:
            name: endpoints
            wrapped: true
          items:
            $ref: '#/components/schemas/ZigbeeEndpoint'
      xml:
        name: ZigbeeEndpointlist

## A Zigbee endpoint with its clusters
    ZigbeeEndpoint:
      required:
        - endpointID
        - clusters
      type: object
      properties:
        endpointID:
```

```
          type: integer
          format: int32
          example: 10
        clusters:
          type: array
          xml:
            name: clusters
            wrapped: true
          items:
            $ref: '#/components/schemas/ZigbeeCluster'
      xml:
        name: ZigbeeEndpoint

## A Zigbee cluster with its attributes
    ZigbeeCluster:
      required:
        - clusterID
        - attributes
      type: object
      properties:
        clusterID:
          type: integer
          format: int32
          example: 0
        attributes:
          type: array
          xml:
            name: attributes
            wrapped: true
          items:
            $ref: '#/components/schemas/ZigbeeAttribute'
      xml:
        name: ZigbeeCluster

## A Zigbee attribute
    ZigbeeAttribute:
      required:
        - attributeID
        - attributeType
      type: object
      properties:
        attributeID:
          type: integer
          format: int32
          example: 1
        attributeType:
          type: integer
          format: int32
          example: 32
```

```
      xml:
        name: ZigbeeAttribute

## Attributes that define a Zigbee attribute
    ZigbeeAttributes:
      required:
        - zigbee
      type: object
      properties:
        zigbee:
          required:
            - endpointID
            - clusterID
            - attributeID
          type: object
          properties:
            endpointID:
              type: integer
              format: int32
              example: 1
            clusterID:
              type: integer
              format: int32
              example: 6
            attributeID:
              type: integer
              format: int32
              example: 16
            type:
              type: integer
              format: int32
              example: 1
      xml:
        name: ZigbeeAttributes

## Attributes that define a Zigbee broadcast
    ZigbeeBroadcast:
      required:
        - zigbee
      type: object
      properties:
        zigbee:
          required:
            - endpointID
            - clusterID
            - attributeID
            - value
          type: object
          properties:
```

```
            endpointID:
              type: integer
              format: int32
              example: 1
            clusterID:
              type: integer
              format: int32
              example: 6
            attributeID:
              type: integer
              format: int32
              example: 16
            type:
              type: integer
              format: int32
              example: 1
            value:
              type: integer
              format: int32
              example: 15
      xml:
        name: ZigbeeBroadcast

# Common objects
## A SCIM object, can be a  device or a group
    Object:
      required:
        - id
      type: object
      properties:
        id:
          type: string
          format: uuid
          example: 12345678-1234-5678-1234-56789abcdef4
        type:
          type: string
          example: device
          enum:
            - device
            - group
        technology:
          type: string
          example: ble
          enum:
            - ble
            - zigbee
      xml:
        name: Object
```

```
## A Service is a device with optional service IDs
    Service:
      allOf:
        - $ref: '#/components/schemas/Object'
      type: object
      properties:
        ble:
          type: object
          properties:
            services:
              type: array
              xml:
                name: services
                wrapped: true
              items:
                $ref: '#/components/schemas/BLEServiceID'
            cached:
              description: |-
                If we can cache information, then device doesn't need
                to be rediscovered before every connected.
              type: boolean
              default: false
            cacheIdlePurge:
              description: cache expiry period, when device allows
              type: integer
              example: 3600 # default 1 hour
            autoUpdate:
              description: |-
                autoupdate services if device supports it (default)
              type: boolean
              example: true
          xml:
            name: ble
      xml:
        name: Service

## A Connection
    Connection:
      allOf:
        - $ref: '#/components/schemas/Service'
      type: object
      properties:
        retries:
          type: integer
          format: int32
          example: 3
        retryMultipleAPs:
          type: boolean
          example: true
```

```
      xml:
        name: Connection

## A specific attribute of an Device
    Attribute:
      allOf:
        - $ref: '#/components/schemas/Object'
      oneOf:
        - $ref: '#/components/schemas/BLEAttributes'
        - $ref: '#/components/schemas/ZigbeeAttributes'
      discriminator:
        propertyName: technology
        mapping:
          ble: '#/components/schemas/BLEAttributes'
          zigbee: '#/components/schemas/ZigbeeAttributes'
      xml:
        name: Attribute

## A value of an attribute of an Device
    AttributeValue:
      allOf:
        - $ref: '#/components/schemas/Attribute'
      required:
        - value
      type: object
      properties:
        value:
          type: string
          format: byte
          example: 0001
        forcedResponse:
          description: do or do not wait for a response?
          type: boolean
          example: true
      xml:
        name: AttributeValue

## A file-based attribute of an Device
    AttributeFile:
      allOf:
        - $ref: '#/components/schemas/Attribute'
      required:
        - filename
      type: object
      properties:
        filename:
          type: string
          example: "firmware.dat"
        chunksize:
```

```
          type: integer
        forcedResponse:
          description: do or do not wait for a response?
          type: boolean
          example: true
      xml:
        name: AttributeFile

## A binary blob-based attribute of an Device
    AttributeBlob:
      allOf:
        - $ref: '#/components/schemas/Attribute'
      required:
        - blob
      type: object
      properties:
        blob:
          type: string
          format: binary
        chunksize:
          type: integer
        forcedResponse:
          description: do or do not wait for a response?
          type: boolean
          example: true
      xml:
        name: AttributeFile

## Conditional read of a value (read until specific value is read)
    AttributeConditional:
      allOf:
        - $ref: '#/components/schemas/Attribute'
      required:
        - value
      type: object
      properties:
        value:
          type: string
          format: byte
          example: 0001
        maxTime:
          description: |-
            maximum time the conditional read should run in seconds
            (default 10 sec, max 60 sec)
          type: integer
        maxRepeat:
          description: |-
           maximum time the conditional read should repeat
           (default 5, max 60)
```

```
          type: integer
        frequency:
          description: |-
            time between reads in seconds (default 1, max 60)
          type: integer
      xml:
        name: AttributeConditional

## A subscription attribute of an Device
    Subscription:
      allOf:
        - $ref: '#/components/schemas/Attribute'
      type: object
      properties:
        topic:
          type: string
          example: enterprise/hospital/pulse_oximeter
        dataFormat:
          description: |-
            how is information decorated? default: timestamped and
            attribute ids.
          type: string
          example: default
          enum:
            - default # decorated with attribute ids
            - timestamped
            - payload
        replay:
          type: boolean
          example: false
          default: false
        forcedAck:
          description: |-
            When not looking at device/attribute support MUST we
            ackhnowledge?
          type: boolean
          example: true
      xml:
        name: Subscription

## A broadcast
    Broadcast:
      allOf:
        - $ref: '#/components/schemas/Object'
      oneOf:
        - $ref: '#/components/schemas/BLEBroadcast'
        - $ref: '#/components/schemas/ZigbeeBroadcast'
      discriminator:
        propertyName: technology
```

```
        mapping:
          ble: '#/components/schemas/BLEBroadcast'
          zigbee: '#/components/schemas/ZigbeeBroadcast'
      required:
        - cycle
      type: object
      properties:
        cycle:
          type: string
          example: single
          enum:
            - single
            - repeat
        # broadcast time in ms
        broadcastTime:
          type: integer
          example: 3000
        # interval between broadcasts in ms
        broadcastInterval:
          type: integer
          example: 500
      xml:
        name: Broadcast

## Topic Name
    TopicName:
      required:
        - topic
      type: object
      properties:
        topic:
          type: string
          example: enterprise/hospital/pulse_oximeter
      xml:
        name: TopicName

## DataStream Topic
    Topic:
      allOf:
        - $ref: '#/components/schemas/TopicName'
      oneOf:
        - $ref: '#/components/schemas/BLETopic'
        - $ref: '#/components/schemas/ZigbeeAttributes'
      discriminator:
        propertyName: technology
        mapping:
          ble: '#/components/schemas/BLETopic'
          zigbee: '#/components/schemas/ZigbeeAttributes'
      type: object
```

```
      properties:
        dataFormat:
          description: |-
            How is information decorated?  Default: device
            and attribute ids.
          type: string
          example: default
          enum:
            - default
            - timestamped
            - payload
        replay:
          type: string
          example: off
          enum:
            - off #default
            - on
        dataApps:
          type: array
          xml:
            name: dataApps
            wrapped: true
          items:
            type: object
            properties:
              dataAppID:
                type: string
                example: https://data-app-1
      xml:
        name: Topic

## FileName
    FileName:
      required:
        - filename
      type: object
      properties:
        filename:
          type: string
          example: "firmware.dat"
      xml:
        name: FileName

## File
    File:
      allOf:
        - $ref: '#/components/schemas/FileName'
      type: object
      properties:
```

```
        file: #file itself is provided
          type: string
          format: binary
        fileURL: #file can be downloaded from a URL
          type: string
          example: "https://domain.com/firmware.dat"
      xml:
        name: File

## Defines an operation in a bulk API
    Operation:
      required:
        - operation
      allOf:
        - type: object
          properties:
            operation:
              type: string
              enum:
               - /extension/connection/create
               - /extension/connection/delete
               - /extension/attribute/read
               - /extension/attribute/write
               - /extension/attribute/write/file
               - /extension/attribute/write/blob
               - /extension/attribute/read/conditional
        - oneOf:
            - $ref: '#/components/schemas/Service'
            - $ref: '#/components/schemas/Attribute'
            - $ref: '#/components/schemas/AttributeValue'
            - $ref: '#/components/schemas/AttributeConditional'
            - $ref: '#/components/schemas/AttributeFile'
            - $ref: '#/components/schemas/AttributeBlob'
          discriminator:
            propertyName: operation
            mapping:
              /extension/connection/create:
                '#/components/schemas/Service'
              /extension/attribute/read:
                '#/components/schemas/Attribute'
              /extension/attribute/read/conditional:
                '#/components/schemas/AttributeConditional'
              /extension/attribute/write:
                '#/components/schemas/AttributeValue'
              /extension/attribute/write/file:
                '#/components/schemas/AttributeFile'
              /extension/attribute/write/blob:
                '#/components/schemas/AttributeBlob'
      xml:
```

```
        name: Operation

## Bulk schema
    Bulk:
      allOf:
        - $ref: '#/components/schemas/Object'
      type: object
      properties:
        autoDisconnect:
          description: |-
            do we automatically disconnect after a RESTful operation?
          type: boolean
          example: true
          default: true
        operations:
          type: array
          xml:
            name: operations
            wrapped: true
          items:
            $ref: '#/components/schemas/Operation'
      xml:
        name: Bulk

# responses
## Baseline success reponse
    Success:
      required:
        - status
      type: object
      properties:
        status:
          type: string
          example: SUCCESS
          enum:
            - SUCCESS
        requestID:
          type: string
          example: 12345678-5678-1234-5578-abcdef1234

    SuccessResponse:
      allOf:
        - $ref: '#/components/schemas/Success'
      type: object
      properties:
        id:
          type: string
          format: uuid
          example: 12345678-1234-5678-1234-56789abcdef4
```

```
## Error 500 application Failure response
    FailureResponse:
      required:
        - status
        - errorCode
      type: object
      properties:
        status:
          type: string
          example: FAILURE
          enum:
            - FAILURE
        reason:
          type: string
          example: Not Found
        errorCode:
          type: integer
          format: int32
          example: 12
        requestID:
          type: string
          example: 12345678-5678-1234-5578-abcdef1234

## Response, success or failure
    Response:
      oneOf:
        - $ref: '#/components/schemas/SuccessResponse'
        - $ref: '#/components/schemas/FailureResponse'
      xml:
        name: Response

## Success response for binding API
    BindingResponse:
      oneOf:
        - $ref: '#/components/schemas/SuccessResponse'
        - $ref: '#/components/schemas/ZigbeeBindingResponse'
        - $ref: '#/components/schemas/FailureResponse'
      xml:
        name: BindingeResponse

## Returning multiple bindings
    MultiBindingsResponse:
      allOf:
        - $ref: '#/components/schemas/Success'
      required:
        - bindings
      type: object
      properties:
```

```
        bindings:
          type: array
          xml:
            name: bindings
            wrapped: true
          items:
            $ref: '#/components/schemas/BindingResponse'
      xml:
        name: MultiBindingsResponse

## Returns Zigbee node & pan ID
    ZigbeeBindingResponse:
      allOf:
        - $ref: '#/components/schemas/SuccessResponse'
      type: object
      properties:
        nodeID:
          type: integer
          format: int32
          example: 65234
        panID:
          type: integer
          format: int32
          example: 48734
      xml:
        name: ZigbeeBindgingResponse

 ## Returns discovered services
    ServiceResponse:
      allOf:
        - $ref: '#/components/schemas/SuccessResponse'
      oneOf:
        - $ref: '#/components/schemas/BLEServiceslist'
        - $ref: '#/components/schemas/ZigbeeEndpointlist'
      xml:
        name: ConnectionResponse

## Response to multiple connections
    MultiConnectionsResponse:
      allOf:
        - $ref: '#/components/schemas/Success'
      required:
        - connections
      type: object
      properties:
        connections:
          type: array
          xml:
            name: connections
```

```
            wrapped: true
          items:
            $ref: '#/components/schemas/Response'
      xml:
        name: MultiConnectionsResponse

## Returns an attribute value
    AttributeValueResponse:
      allOf:
        - $ref: '#/components/schemas/SuccessResponse'
      required:
       - value
      type: object
      properties:
        value:
          type: string
          example: 01
          format: byte
      xml:
        name: AttributeValueResponse

## Returns a topic
    TopicResponse:
      allOf:
        - $ref: '#/components/schemas/SuccessResponse'
      required:
        - topic
      type: object
      properties:
        topic:
          type: string
          example: enterprise/hospital/pulse_oximeter
      xml:
        name: TopicResponse

## Returning multiple topics
    MultiTopicsResponse:
      allOf:
        - $ref: '#/components/schemas/Success'
      required:
        - topics
      type: object
      properties:
        topics:
          type: array
          xml:
            name: topics
            wrapped: true
          items:
```

```
            $ref: '#/components/schemas/TopicName'
      xml:
        name: MultiTopicsResponse

## Returning multiple subscriptions
    MultiSubscriptionResponse:
      allOf:
        - $ref: '#/components/schemas/Success'
      required:
        - subscriptions
      type: object
      properties:
        subscriptions:
          type: array
          xml:
            name: subscriptions
            wrapped: true
          items:
            $ref: '#/components/schemas/Subscription'
      xml:
        name: MultiSubscriptionResponse

## Returns a file name
    FileResponse:
      allOf:
        - $ref: '#/components/schemas/SuccessResponse'
      required:
        - filename
      type: object
      properties:
        filename:
          type: string
          example: "firmware.dat"
      xml:
        name: FileResponse

## Returning multiple file names
    MultiFileResponse:
      allOf:
        - $ref: '#/components/schemas/Success'
      required:
        - filenames
      type: object
      properties:
        filenames:
          type: array
          xml:
            name: filenames
            wrapped: true
```

```
          items:
            $ref: '#/components/schemas/FileName'
      xml:
        name: MultiFileResponse

## Multiple returns for a bulk operation
    BulkResponse:
      allOf:
        - $ref: '#/components/schemas/SuccessResponse'
      type: object
      properties:
        operations:
          type: array
          xml:
            name: operations
            wrapped: true
          items:
            $ref: '#/components/schemas/OperationResponse'
      xml:
        name: BulkResponse

## Return for an operation
    OperationResponse:
      required:
        - operation
      allOf:
        - type: object
          properties:
            operation:
              type: string
              enum:
               - /connectivity/connection/create
               - /connectivity/connection/delete
               - /extension/attribute/read
               - /extension/attribute/write
               - /extension/attribute/write/file
               - /extension/attribute/write/blob
               - /extension/attribute/read/conditional
        - oneOf:
          - $ref: '#/components/schemas/ServiceResponse'
          - $ref: '#/components/schemas/SuccessResponse'
          - $ref: '#/components/schemas/AttributeValueResponse'
          discriminator:
            propertyName: operation
            mapping:
              /connectivity/connection/create:
                '#/components/schemas/ServiceResponse'
              /connectivity/connection/delete:
                '#/components/schemas/SuccessResponse'
```

```
              /extension/attribute/read:
                '#/components/schemas/AttributeValueResponse'
              /extension/attribute/write:
                '#/components/schemas/AttributeValueResponse'
              /extension/attribute/write/file:
                '#/components/schemas/SuccessResponse'
              /extension/attribute/write/blob:
                '#/components/schemas/SuccessResponse'
              /extension/attribute/read/conditional:
                '#/components/schemas/AttributeValueResponse'
      xml:
        name: Operation

 # API key authorization
  securitySchemes:
    ApiKeyAuth:
      type: apiKey
      in: header
      name: X-API-KEY
# Apply the API key globally to all operations
security:
  - ApiKeyAuth: []

<CODE ENDS>
```

Figure 6

Appendix B. Protobuf definition

The following non-normative protocol buffer definition is provide for convenience of the implementor.

```
<CODE BEGINS> file "data_app.proto"

syntax = "proto3";

option java_package = "org.ietf.nipc.proto";
option java_multiple_files = true;

package nipc;

message DataSubscription {
    optional string device_id = 1;
    bytes data = 2;

    oneof subscription {
        BLESubscription ble_subscription = 3;
        BLEAdvertisement ble_advertisement = 4;
        ZigbeeSubscription zigbee_subscription = 5;
        RawPayload raw_payload = 6;
        BLEConnectionStatus ble_connection_status = 7;
    }

    message BLESubscription {
        optional string service_uuid = 1;
        optional string characteristic_uuid = 2;
    }

    message BLEAdvertisement {
        string mac_address = 1;
        optional int32 rssi = 2;
    }

    message ZigbeeSubscription {
        optional int32 endpoint_id = 1;
        optional int32 cluster_id = 2;
        optional int32 attribute_id = 3;
        optional int32 attribute_type = 4;
    }

    message BLEConnectionStatus {
        string mac_address = 1;
        bool connected = 2;
        optional int32 reason = 3;
    }

    message RawPayload {
        optional string context_id = 1;
    }
}
```

<CODE ENDS>

Figure 7

Authors' Addresses

Bart Brinckman
Cisco Systems
Brussels
Belgium

Email: bbrinckm@cisco.com

Rohit Mohan
Cisco Systems
170 West Tasman Drive
San Jose, 95134
United States of America

Email: rohitmo@cisco.com

Braeden Sanford
Philips
Cambridge,
United States of America

Email: braeden.sanford@philips.com