Translator FIWARE NGSIv2/HTTP/TLS

Interface Design Description

Service ID: *"translator-fiware"*

Abstract
This document describes the Translator FIWARE service IDD for NGSIv2/HTTP/TLS.

ARTEMIS Innovation Pilot Project: Arrowhead
THEME [SP1-JTI-ARTEMIS-2012-AIPP4 SP1-JTI-ARTEMIS-2012-AIPP6]
[Production and Energy System Automation Intelligent-Built environment and urban infrastructure for sustainable and friendly cities]

Contents

1 Overview

This document describes the NGSIv2/HTTP/TLS variant of the Translator FIWARE Eclipse Arrowhead service. The Translator FIWARE service is used to exchange messages between a data producers and consumers within a FIWARE network. Tranlator FIWARE service includes two components a FIWARE interface, and a Plugin Service to generate dinamically services.

This document exists as a complement to the *Translator FIWARE – Service Translator* (Translator FIWARE SD) document. For further details about how this service is meant to be used, please consult that document. The rest of this document describes how to realize the Translator FIWARE service using HTTP [1], TLS [2] and NGSIv2 [3], in terms of its interfaces (Section 2) and its information model (Section 3).

2 Service Interfaces

This section lists the interfaces that must be exposed by Translator FIWARE service. In particular, each subsection first names the HTTP method and path used to call the interface, after which it names an abstract interface from the Translator FIWARE service's SD document, as well as input and output types. All interfaces in this section respond with the HTTP status code `200 OK` if called successfully, unless otherwise is stated.

2.1 GET /translator/plugin/service/$entityId/$serviceName

Interface: PluginGetEntityValue
Output: Object

Called to request a service value with Plugin Service, as exemplified in Listing 1, 2 and 3.

```
GET /translator/plugin/service/test-device/temperature HTTP/1.1 ACCEPT=TEXT_PLAIN_VALUE

23.7
```

Listing 1: Response in Text Plain.

```
GET /translator/plugin/service/test-device/temperature HTTP/1.1 ACCEPT=APPLICATION_SENML

[
  {
    "bt": 1507036983434,
    "v": 23.7,
    ...
  }
]
```

Listing 2: Response in SenML.

```
GET /translator/plugin/service/test-device/temperature HTTP/1.1 ACCEPT=APPLICATION_JSON

{
  "time": 1507036983434,
  "value": 23.7,
  ...
}
```

Listing 3: Response in JSON.

Code	Type	Description
200	OK	No error
400	BAD REQUEST	Bad request
401	UNAUTHORIZED	No valid authorization
404	NOT FOUND	Resource not found
415	UNSUPPORTED MEDIA TYPE	Wrong requested Media Type
500	INTERNAL SERVER ERROR	Server error, etc

Table 1: PluginGetEntityValue responses

2.2 GET /translator/v2

Interface: **FiwareGetIt**
Output: **FiwareUrlServices**

Called to request the URLs of the FIWARE services supported, as exemplified in Listing 4.

```
GET /translator/v2 HTTP/1.1

{
  "entities_url": "/translator/v2/entities",
  "types_url": "/translator/v2/types"
}
```

Listing 4: A FiwareUrlServices.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Server error, etc

Table 2: FiwareGetIt responses

2.3 GET /translator/v2/entities

Interface: **FiwareListEntities**
Input: **id,type,idPattern,typePattern,q,mq,georel,geometry,coords,limit,offset,attrs,metadata,orderBy,options**
Output: **FiwareEntity**

Called to request the URLs of the FIWARE services supported, as exemplified in Listing 5.

```
GET /translator/v2/entities HTTP/1.1

[
  {
    "type": "TempSensor",
    "id": "SensorTest1",
    "temperature": {
      "value": 35.6,
      "type": "Number",
      "metadata": {}
    }
  },
  {
    "type": "TempSensor",
    "id": "SensorTest2",
    "temperature": {
      "value": 22.5,
      "type": "Number",
      "metadata": {}
    }
  },
  {
    "type": "Vehicle",
    "id": "D10S-KK",
    "speed": {
      "value": 100,
      "type": "number",
      "metadata": {
        "accuracy": {
          "value": 2,
          "type": "Number"
        },
        "timestamp": {
          "value": "2015-06-04T07:20:27.378Z",
          "type": "DateTime"
        }
      }
    }
  }
]
```

Listing 5: A FiwareEntity List.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Server error, etc

Table 3: FiwareListEntities responses

2.4 POST /translator/v2/entities

Interface: **FiwareCreateEntity**
Input: **FiwareEntity**

Called to create a new FIWARE Entity, as exemplified in Listing 6.

```
POST /translator/v2/entities HTTP/1.1
```

Listing 6: A FiwareEntity creation.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
422	UNPROCESSABLE ENTITY	Improssible to process the entity
500	INTERNAL SERVER ERROR	Server error, etc

Table 4: FiwareCreateEntity responses

2.5 GET /translator/v2/entities/$id

Interface: **FiwareRetrieveEntity**
Input: **id,type,attrs,metadata,options**
Output: **FiwareEntity**

Called to request an specific FIWARE Entity, as exemplified in Listing 7.

```
GET /translator/translator/v2/entities/SensorTest2 HTTP/1.1

{
  "type": "TempSensor",
  "id": "SensorTest2",
  "temperature": {
    "value": 22.5,
    "type": "Number",
    "metadata": {}
  }
}
```

Listing 7: A FiwareEntity.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Server error, etc

Table 5: FiwareRetrieveEntity responses

2.6 GET /translator/v2/entities/$id/attrs

Interface: FiwareRetrieveEntityAttributes
Input: id, type, attrs, metadata, options
Output: Object

Called to request the attributes of an specific FIWARE Entity, as exemplified in Listing 8.

```
GET /translator/v2/entities/sensor-test3/attrs HTTP/1.1

{
  "temperature": {
    "value": 21.7,
    "type": "Number"
  },
  "humidity": {
    "value": 60,
    "type": "Number"
  },
  "location": {
    "value": "41.3763726, 2.1864475",
    "type": "geo:point",
    "metadata": {
      "crs": {
        "value": "WGS84",
        "type": "Text"
      }
    }
  }
}
```

Listing 8: A JSON Object attirbutes example.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Server error, etc

Table 6: FiwareRetrieveEntityAttributes responses

2.7 POST /translator/v2/entities/$id/attrs

Interface: FiwareUpdateAppendEntityAttributes
Input: id, type, attrs, metadata, options, Object

Called to add or update the attributes of an specific FIWARE Entity, as exemplified in Listing 9.

```
POST /translator/v2/entities/sensor-test3/attrs HTTP/1.1
```

Listing 9: A JSON Object attirbutes example.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Server error, etc

Table 7: FiwareUpdateAppendEntityAttributes responses

2.8 DELETE /v2/entities/$id

Interface: **FiwareRemoveEntity**
Input: **id, type**

Called to remove an specific FIWARE Entity, as exemplified in Listing 10.

```
DELETE /v2/entities/sensor-test3 HTTP/1.1
```

Listing 10: A FiwareRemoveEntity request example.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Server error, etc

Table 8: FiwareRemoveEntity responses

2.9 GET /translator/v2/types

Interface: **FiwareListEntityTypes**
Input: **limit, offset, options**
Output: **Object**

Called to request a list of registered FIWARE types, as exemplified in Listing 11.

```
GET /translator/v2/types HTTP/1.1

[
  {
    "type": "TempSensor",
    "attrs": {
      "temperature": {
        "types": [
          "urn:phenomenum:temperature"
        ]
      }
    },
    "count": 3
  },
  {
    "type": "Vehicle",
    "attrs": {
      "speed": {
        "types": [
          "Number"
        ]
      },
      "fuel": {
        "types": [
          "gasoline",
          "diesel"
        ]
      }
    },
    "count": 1
  }
]
```

Listing 11: A JSON Object types list example.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Server error, etc

Table 9: FiwareListEntityTypes responses

2.10 GET /translator/v2/types/$type

Interface: FiwareRetrieveEntityType
Input: type
Output: Object

Called to request a specific registered FIWARE types, as exemplified in Listing 12.

```
GET /translator/v2/types/TempSensor HTTP/1.1

{
  "attrs": {
    "temperature": {
      "types": [
        "urn:phenomenum:temperature"
      ]
    }
  },
  "count": 3
}
```

Listing 12: A JSON Object type example.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Server error, etc

Table 10: FiwareRetrieveEntityType responses

3 Information Model

The default payload type is JSON-encoded. The response to a GET request is a simple HTTP status code (Created/OK – request was a success, No Content – request had no effect). For the Push interface the content-type must be set to 'application/json'. For Fetch, the response content-type is 'application/json'.

3.1 struct FiwareUrlServices

All messages must be encoded using JSON.

Object Field	Value Type	Description
entities_url	String	Entities URL path.
types_url	String	Types URL path.
subscriptions_url	String	Subscriptions URL path.
registrations_url	String	Registrations URL path.

3.2 struct FiwareEntity

All messages must be encoded using JSON.

Object Field	Value Type	Description
id	String	Id of the Entity.
type	String	Type of the Entity.
others	Any	Any other parameters.

3.3 Primitives

Types and structures mentioned throughout this document that are assumed to be available to implementations of this service. The concrete interpretations of each of these types and structures must be provided by any IDD document claiming to implement this service.

Type	Description
id	A comma-separated list of elements. Retrieve entities whose ID matches one of the elements in the list..
type	Comma-separated list of elements. Retrieve entities whose type matches one of the elements in the list.
idPattern	A correctly formated regular expression. Retrieve entities whose ID matches the regular expression. Incompatible with id.
typePattern	A correctly formated regular expression. Retrieve entities whose type matches the regular expression. Incompatible with type.
q	A query expression, composed of a list of statements separated by ;.
mq	A query expression for attribute metadata, composed of a list of statements separated by ;.
georel	Spatial relationship between matching entities and a reference shape.
geometry	Geografical area to which the query is restricted.
coords	List of latitude-longitude pairs of coordinates separated by ';'.
limit	Limits the number of entities to be retrieved.
offset	Establishes the offset from where entities are retrieved.
attrs	Comma-separated list of attribute names whose data are to be included in the response. The attributes are retrieved in the order specified by this parameter. If this parameter is not included, the attributes are retrieved in arbitrary order.
metadata	A list of metadata names to include in the response.
orderBy	Criteria for ordering results.
options	Options dictionary.
URL path	String of the Path of a URL.
Any	Any JSON Element (JSON Object or JSON Array).
Object	JSON Object.

3.4 Canonical Forms

Values conforming to some of the types in this document will have to be hashed to produce necessary identifiers. This requires that all relevant values can be presented in a canonical form. The canonical form of all types described in this document are produced by encoding them in JSON, but with the following restrictions:

1. UTF-8 encoding must be used.
2. No insignificant whitespace may be used, as defined in [4].
3. No String escapes may be used. String are to be treated as raw UTF-8 byte arrays enclosed with double quotes, even if they contain illegal UTF-8 characters.
4. Integer Numbers must
   (a) not have a leading minus sign if zero,

   (b) have no fraction or exponent, and
   (c) must not contain any leading zeroes, unless exactly 0.

5. All other Numbers must
   (a) contain an integral of exactly 1 non-zero digit,
   (b) have a fraction, unless it is effectively 0,
   (c) not have trailing fraction zeroes,
   (d) have an exponent, unless it is effectively 0,
   (e) use capital ”E” as exponent marker,
   (f) not have a leading exponent plus sign,
   (g) not have any trailing exponent zeroes.
6. Object pairs must be provided in the same order as they are listed in their type definitions. If no type definition exists, due to the Object being interpreted as being an arbitrary mapping between keys and value, the pairs must be sorted in ascending alphabetical order by their keys.
7. Object pairs with values of the Null type must be omitted.

Note that the Numbers `0`, `-1` and `12500` are canonical integers, while `-0`, `0.0` and `002` are not. Furthermore, the decimal Numbers `5`, `1.025E3` and `4E-9` are canonical, while `7.0`, `12E4`, `7.10`, `4E+9`, `6.2E02` and `3E0` are not. Also note that some integer numbers are represented in exactly the same way as decimal numbers without fractions and exponents.

4 References

[1] R. Fielding and J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing,” RFC 7230, 2018, RFC Editor. [Online]. Available: https://doi.org/10.17487/RFC7230

[2] E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3,” RFC 8446, 2018, RFC Editor. [Online]. Available: https://doi.org/10.17487/RFC8446

[3] “Next Generation Service Interface - NGSI,” Online, 2019, . [Online]. Available: https://www.opemmobilealliance.org/

[4] T. Bray, “The JavaScript Object Notation (JSON) Data Interchange Format,” RFC 7159, 2014, RFC Editor. [Online]. Available: https://doi.org/10.17487/RFC7159

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2019-03-27	1.0	Initial	Pablo Puñal Pereira
2	2019-05-02	1.1	Models and Interfaces update	Pablo Puñal Pereira
3	2020-12-11	2.0	Template Update	Pablo Puñal Pereira

5.2 Quality Assurance

No.	Date	Version	Approved by
1			