OCF Bridging Specification

VERSION 2.0.2 | April 2019

OPEN CONNECTIVITY FOUNDATION™

CONTACT admin@openconnectivity.org

LEGAL DISCLAIMER

CONTENTS

Figures

Tables

1 Scope

This document specifies a framework for translation between OCF Devices and other ecosystems, and specifies the behaviour of a Bridging Function that exposes servers in non-OCF ecosystem to OCF Clients and/or exposes OCF Servers to clients in non-OCF ecosystem. Translation per specific Device is left to other documents (deep translation). This document provides generic requirements that apply unless overridden by a more specific document.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

AllJoyn About Interface Specification, *About Feature Interface Definitions*, Version 14.12
https://github.com/alljoyn/extras-webdocs/blob/master/docs/learn/core/about-announcement/interface.md

AllJoyn Configuration Interface Specification, *Configuration Interface Definition*, Version 14.12
https://github.com/alljoyn/extras-webdocs/blob/master/docs/learn/core/configuration/interface.md

D-Bus Specification, *D-Bus Specification*
https://dbus.freedesktop.org/doc/dbus-specification.html

IEEE 754, *IEEE Standard for Floating-Point Arithmetic*, August 2008
http://ieeexplore.ieee.org/servlet/opac?punumber=4610933

IETF RFC 4122, *A Universally Unique Identifier (UUID) URN Namespace*, July 2005
https://www.rfc-editor.org/info/rfc4122

IETF RF 4648, *The Base16, Base32 and Base64 Data Encodings*, October 2006
https://www.rfc-editor.org/info/rfc4648

IETF RFC 6973, *Privacy Considerations for Internet Protocols*, July 2013
https://www.rfc-editor.org/info/rfc6973

IETF RFC 7159, *The JavaScript Object Notation (JSON) Data Interchange Format*, March 2014
https://www.rfc-editor.org/info/rfc7159

ISO/IEC 30118-1:2018 Information technology -- Open Connectivity Foundation (OCF) Specification -- Part 1: Core specification
https://www.iso.org/standard/53238.html
Latest version available at: https://openconnectivity.org/specs/OCF_Core_Specification.pdf

ISO/IEC 30118-2:2018 Information technology -- Open Connectivity Foundation (OCF) Specification -- Part 2: Security specification
https://www.iso.org/standard/74239.html
Latest version available at: https://openconnectivity.org/specs/OCF_Security_Specification.pdf

ISO/IEC 30118-6:2018 Information technology -- Open Connectivity Foundation (OCF) Specification -- Part 6: Resource to AllJoyn interface mapping specification
https://www.iso.org/standard/74243.html
Latest version available at:
https://openconnectivity.org/specs/OCF_Resource_to_AllJoyn_Interface_Mapping.pdf

JSON Schema Core, *JSON Schema: core definitions and terminology*, January 2013
http://json-schema.org/latest/json-schema-core.html

JSON Schema Validation, *JSON Schema: interactive and non-interactive validation*, January 2013
http://json-schema.org/latest/json-schema-validation.html

JSON Hyper-Schema, *JSON Hyper-Schema: A Vocabulary for Hypermedia Annotation of JSON*, October 2016
http://json-schema.org/latest/json-schema-hypermedia.html

OpenAPI Specification, Version 2.0
https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md

OCF Resource to oneM2M Module Class Mapping, *Open Connectivity Foundation Resource to oneM2M Module Class Mapping Specification*, version 2.0.2
Available at:
https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification_v2.0.2.pdf
Latest version available at:
https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification.pdf

3 Terms, definitions, and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 30118-1:2018 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at https://www.iso.org/obp
- IEC Electropedia: available at http://www.electropedia.org/

3.1.1
Asymmetric Client Bridge
an asymmetric client bridge exposes another ecosystem clients into the OCF ecosystem as Virtual OCF Clients (3.1.2). This is equivalent to exposing OCF Servers (3.1.15) into the other ecosystem. How this is handled in each ecosystem is specified on a per ecosystem basis in this document.

3.1.2
Asymmetric Server Bridge
an asymmetric server bridge exposes another ecosystem devices into the OCF ecosystem as Virtual OCF Servers (3.1.26). How this is handled in each ecosystem is specified on a per ecosystem basis in this document.

3.1.3
Bridge
OCF Device that has a Device Type of "oic.d.bridge", provides information on the set of Virtual OCF Devices (3.1.24) that are resident on the same Bridge Platform.

3.1.4
Bridge Platform
Entity on which the Bridge (3.1.2) and Virtual OCF Devices (3.1.25) are resident

3.1.5
Bridged Client

logical entity that accesses data via a Bridged Protocol (3.1.5). For example, an AllJoyn Consumer application is a Bridged Client

3.1.6
Bridged Device

Bridged Client (3.1.3) or Bridged Server (3.1.8).

3.1.7
Bridged Protocol

another protocol (e.g., AllJoyn) that is being translated to or from OCF protocols

3.1.8
Bridged Resource

represents an artefact modelled and exposed by a Bridged Protocol (3.1.5), for example an AllJoyn object is a Bridged Resource.

3.1.9
Bridged Resource Type

schema used with a Bridged Protocol (3.1.5), for example AllJoyn Interfaces are Bridged Resource Types.

3.1.10 Bridged Server

logical entity that provides data via a Bridged Protocol (3.1.5), for example an AllJoyn Producer is a Bridged Server. More than one Bridged Server can exist on the same physical platform.

3.1.11
Bridging Function

Logic resident on the Bridge Platform (3.1.4) that performs that protocol mapping between OCF and the Bridged Protocol (3.1.7); a Bridge Platform (3.1.4) may contain multiple Bridging Functions dependent on the number of Bridged Protocols (3.1.7) supported.

3.1.12
OCF Bridge Device

OCF Device (3.1.11) that can represent devices that exist on the network but communicate using a Bridged Protocol (3.1.5) rather than OCF protocols.

3.1.13
OCF Client

logical entity that accesses an OCF Resource (3.1.12) on an OCF Server (3.1.15), which might be a Virtual OCF Server (3.1.26) exposed by the OCF Bridge Device (3.1.9)

3.1.14
OCF Device

logical entity that assumes one or more OCF roles (OCF Client (3.1.10), OCF Server (3.1.15). More than one OCF Device can exist on the same physical platform.

3.1.15
OCF Resource

represents an artefact modelled and exposed by the OCF Framework

3.1.16
OCF Resource Property

significant aspect or notion including metadata that is exposed through the OCF Resource (3.1.12)

3.1.17
OCF Resource Type
OCF Resource Property (3.1.13) that represents the data type definition for the OCF Resource (3.1.12)

3.1.18
OCF Server
logical entity with the role of providing resource state information and allowing remote control of its resources

3.1.19
oneM2M Application
In an OCF-oneM2M asymmetric bridge environment, the oneM2M application represents the oneM2M control point (i.e. client) being mapped to a virtual OCF client.

3.1.20
Symmetric, Asymmetric Bridging
in symmetric bridging, a bridge device exposes OCF Server(s) (3.1.15) to another ecosystem and exposes other ecosystem's server(s) to OCF. In asymmetric bridging, a bridge device exposes OCF Server(s) (3.1.15) to another ecosystem or exposes another ecosystem's server(s) to OCF, but not both.

3.1.21
Virtual Bridged Client
logical representation of an OCF Client (3.1.10), which an OCF Bridge Device (3.1.9) exposes to Bridged Servers (3.1.8).

3.1.22
Virtual Bridged Server
logical representation of an OCF Server (3.1.15), which an OCF Bridge Device (3.1.9) exposes to Bridged Clients (3.1.3).

3.1.23
Virtual OCF Client
logical representation of a Bridged Client (3.1.3), which an OCF Bridge Device (3.1.9) exposes to OCF Servers (3.1.15)

3.1.24
Virtual OCF Device
Virtual OCF Client (3.1.23) or Virtual OCF Server (3.1.26).

3.1.25
Virtual OCF Resource
logical representation of a Bridged Resource (3.1.6), which an OCF Bridge Device (3.1.9) exposes to OCF Clients (3.1.10)

3.1.26
Virtual OCF Server
logical representation of a Bridged Server (3.1.8), which an OCF Bridge Device (3.1.9) exposes to OCF Clients (3.1.10).

3.2 Abbreviated terms

3.2.1
CRUDN
Create, Read, Update, Delete, and Notify

3.2.2
CSV
Comma separated value

4 Document conventions and organization

4.1 Conventions

In this document a number of terms, conditions, mechanisms, sequences, parameters, events, states, or similar terms are printed with the first letter of each word in uppercase and the rest lowercase (e.g., Network Architecture). Any lowercase uses of these words have the normal technical English meaning

4.2 Notation

In this document, features are described as required, recommended, allowed or DEPRECATED as follows:

Required (or shall or mandatory).

– These basic features shall be implemented to comply with OIC Core Architecture. The phrases “shall not”, and "PROHIBITED" indicate behaviour that is prohibited, i.e. that if performed means the implementation is not in compliance.

Recommended (or should).

– These features add functionality supported by OIC Core Architecture and should be implemented. Recommended features take advantage of the capabilities OIC Core Architecture, usually without imposing major increase of complexity. Notice that for compliance testing, if a recommended feature is implemented, it shall meet the specified requirements to be in compliance with these guidelines. Some recommended features could become requirements in the future. The phrase "should not" indicates behaviour that is permitted but not recommended.

Allowed (or allowed).

– These features are neither required nor recommended by OIC Core Architecture, but if the feature is implemented, it shall meet the specified requirements to be in compliance with these guidelines.

– Conditionally allowed (CA)The definition or behaviour depends on a condition. If the specified condition is met, then the definition or behaviour is allowed, otherwise it is not allowed.

Conditionally required (CR)

– The definition or behaviour depends on a condition. If the specified condition is met, then the definition or behaviour is required. Otherwise the definition or behaviour is allowed as default unless specifically defined as not allowed.

DEPRECATED

– Although these features are still described in this document, they should not be implemented except for backward compatibility. The occurrence of a deprecated feature during operation of an implementation compliant with the current document has no effect on the implementation’s operation and does not produce any error conditions. Backward compatibility may require that a feature is implemented and functions as specified but it shall never be used by implementations compliant with this document.

Strings that are to be taken literally are enclosed in "double quotes".

Words that are emphasized are printed in *italic*.

5 Bridge Platform

5.1 Introduction

This clause describes the functionality of a Bridge Platform; such a device is illustrated in Figure 1.

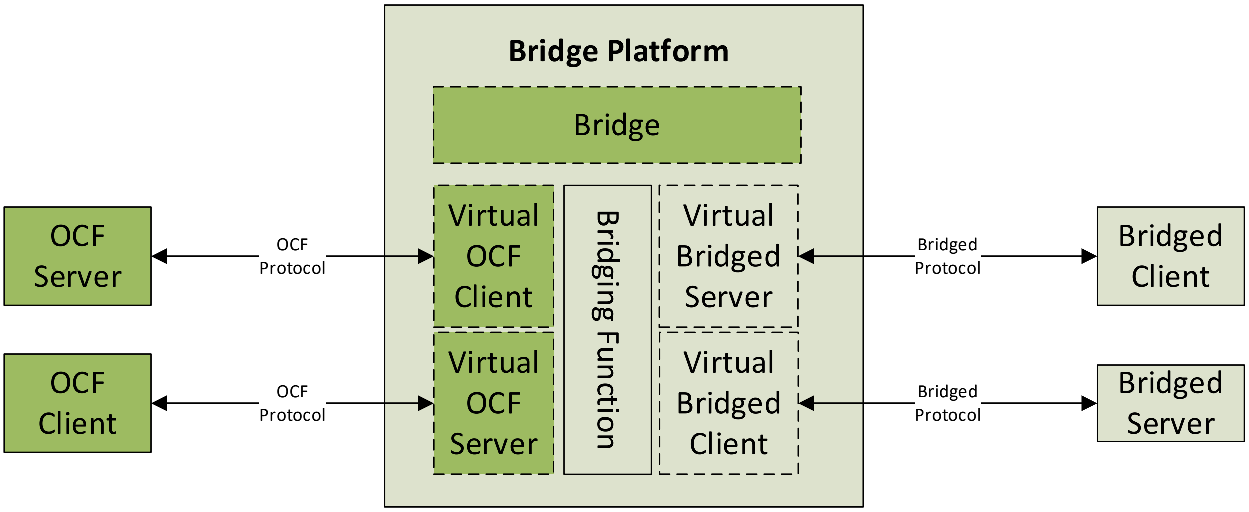

Figure 1 – Bridge Platform components

A Bridge Platform enables the representation of one or more Bridged Devices as Virtual OCF Devices (VODs) on the network and/or enables the representation of one or more OCF Devices as Virtual OCF Devices using another protocol on the network. The Bridged Devices themselves are out of the scope of this document. The only difference between a native OCF Device and a VOD from the perspective of an OCF Client is the inclusion of "oic.d.virtual" in the "rt" of "/oic/d" of the VOD.

A Bridge Platform exposes a Bridge Device which is an OCF Device with a Device Type of "oic.d.bridge". This provides to an OCF Client an explicit indication that the discovered Device is performing a bridging function. This is useful for several reasons; 1) when establishing a home network, the Client can determine that the bridge is reachable and functional when no bridged devices are present, 2) allows for specific actions to be performed on the bridge considering the known functionality a bridge supports, 3) allows for explicit discovery of all devices that are serving a bridging function which benefits trouble shooting and maintenance actions on behalf of a user. When such a device is discovered the exposed Resources on the OCF Bridge Device describe other devices. For example, as shown in Figure 2.

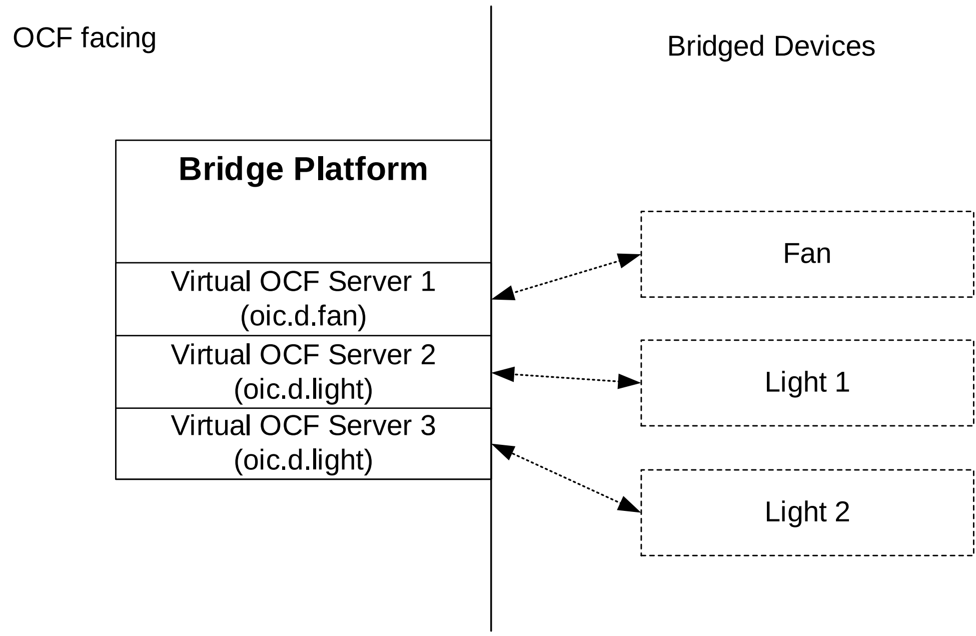

Figure 2 – Schematic overview of a Bridge Platform bridging non-OCF devices

It is expected that the Bridge Platform creates a set of devices during the start-up of the Bridge Platform, these being the Bridge and any known VODs. The exposed set of VODs can change as Bridged Devices are added or removed from the bridge. The adding and removing of Bridged Devices is implementation dependent.

5.2 Symmetric vs. asymmetric bridging

There are two kinds of bridging: Symmetric, Asymmetric. In symmetric bridging, a bridge device exposes OCF server(s) to another ecosystem and exposes other ecosystem's server(s) to OCF. In asymmetric bridging, a bridge device exposes OCF server(s) to another ecosystem or exposes another ecosystem's server(s) to OCF, but not both. The former case is called an Asymmetric Server Bridge (see Figure 3), the latter case is called an Asymmetric Client Bridge (see Figure 4)

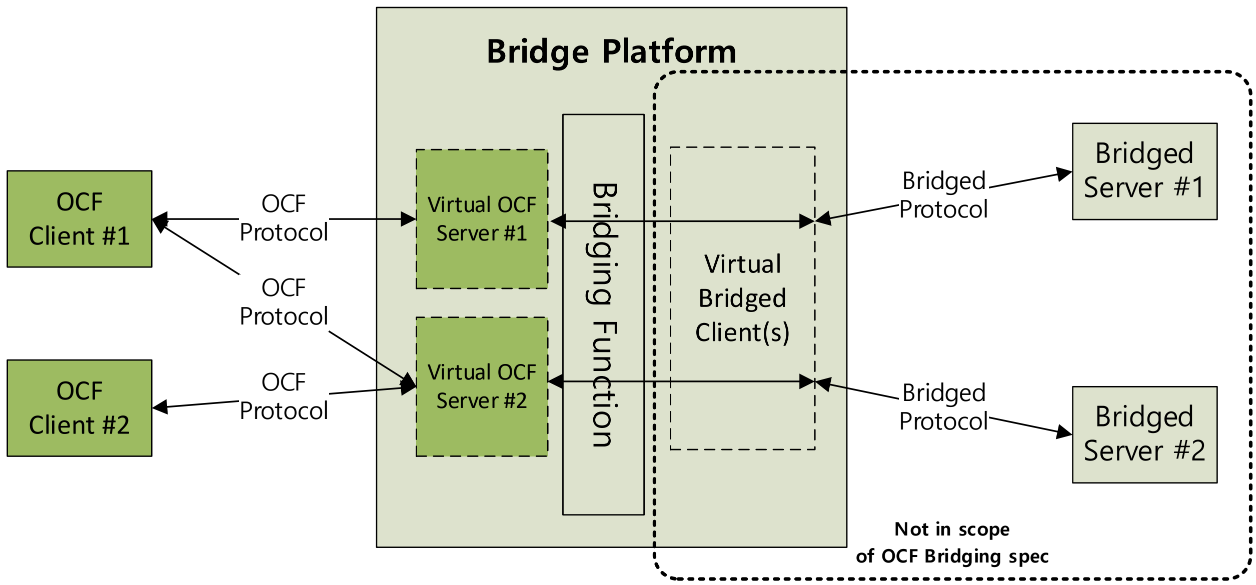

Figure 3 – Asymmetric server bridge

In Figure 3 each Bridged Server is exposed as a Virtual OCF Server to OCF side. These Virtual OCF Servers are same as normal OCF Servers except that they have additional rt value ("oic.d.virtual") for "/oic/d". The details of the Virtual Bridged Client are not in scope of this document.

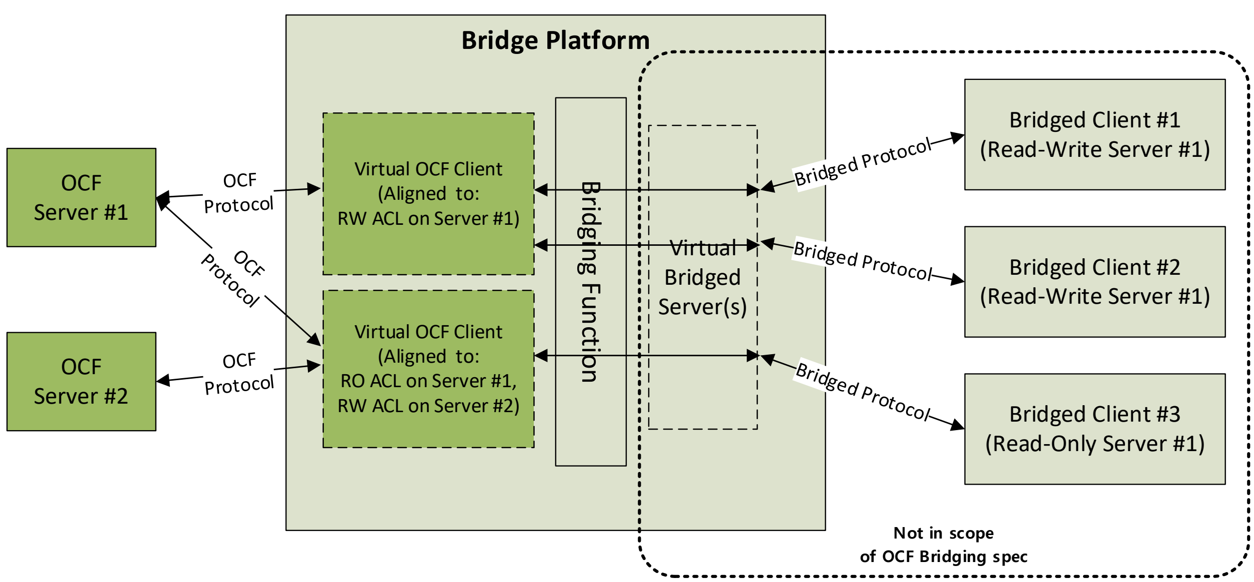

Figure 4 – Asymmetric client bridge

Figure 4 shows that each access to the OCF Server is modelled as a Virtual OCF Client. Those accesses can be aggregated if their target OCF servers and access permissions are same, therefore a Virtual OCF Client can tackle multiple Bridged Clients.

5.3 General requirements

5.3.1 For Asymmetric Bridging

A VOD shall have a Device Type that contains "oic.d.virtual". This allows Bridge Platforms to determine if a device is already being translated when multiple Bridge Platforms are present.

Each Bridged Server shall be exposed as a separate Virtual OCF Device, with its own OCF Endpoint, and set of mandatory Resources (as defined in ISO/IEC 30118-1:2018 and ISO/IEC 30118-2:2018). Discovery of a VOD is the same as for an ordinary OCF Device; that is the VOD shall respond to multicast discovery requests. This allows platform-specific, device-specific, and resource-specific fields to all be preserved across translation.

The Bridge Introspection Device Data (IDD) provides information for the Resources exposed by the Bridge only. Each VOD shall expose an instance of "oic.wk.introspection" which provides a URL to an IDD for the specific VOD.

5.3.2 For Symmetric Bridging

In addition to the requirements mentioned in 5.3.1, Symmetric Bridging shall satisfy following requirements.

The Bridge Platform shall check the protocol-independent UUID ("piid" in OCF) of each device and shall not advertise back into a Bridged Protocol a device originally seen via that Bridged Protocol. The Bridge Platform shall stop translating any Bridged Protocol device exposed in OCF via another Bridge Platform if the Bridge Platform sees the device via the Bridged Protocol. Similarly, the Bridge Platform shall not advertise an OCF Device back into OCF, and the Bridge Platform shall stop translating any OCF device exposed in the Bridged Protocol via another Bridge Platform if the Bridge Platform sees the device via OCF. These require that the Bridge Platform can determine when a device is already being translated. A VOD shall be indicated on the OCF Security Domain with a Device Type of "oic.d.virtual". How a Bridge Platform determines if a device is already being translated on a non-OCF Security Domain is described in the protocol-specific clauses (e.g. clause 6).

The Bridge Platform shall detect duplicate VODs (with the same protocol-independent UUID) present in a network and shall not create more than one corresponding virtual device as it translates those duplicate devices into another network.

5.4 Resource discovery

A Bridge Platform shall detect devices that arrive and leave the Bridged network or the OCF Security Domain. Where there is no pre-existing mechanism to reliably detect the arrival and departure of devices on a network, a Bridge Platform shall periodically poll the network to detect arrival and departure of devices, for example using COAP multicast discovery (a multicast RETRIEVE of "/oic/res") in the case of the OCF Security Domain. Bridge Platform implementations are encouraged to use a poll interval of 30 seconds plus or minus a random delay of a few seconds.

An Bridge Platform and any exposed VODs shall each respond to network discovery commands. The response to a RETRIEVE on "/oic/res" shall only include the devices that match the RETRIEVE request.

The resource reference determined from each Link exposed by "/oic/res" on the Bridge or on a VOD shall be unique. The Bridge and the VODs shall meet the requirements defined in ISO/IEC 30118-1:2018 for population of the Properties and Link parameters in "/oic/res".

For example, if a Bridge exposes VODs for the fan and lights shown in Figure 2, and an OCF Client performs a discovery request with a content format of "application/vnd.ocf+cbor", there will be four discrete responses, one for the Bridge, one for the virtual fan Device, and two for the virtual light

Devices. Note that what is returned is not in the JSON format but in a suitable encoding as defined in ISO/IEC 30118-1:2018.

```
 Response from the Bridge:
```


```
[
 {
   "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
   "href": "/oic/res",
   "rel": "self",
   "rt": ["oic.wk.res"],
   "if": ["oic.if.ll", "oic.if.baseline"],
   "p": {"bm": 3},
   "eps": [{"ep": "coap://[2001:db8:a::b1d4]:55555"},
           {"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
 },
 {
   "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
   "href": "/oic/d",
   "rt": ["oic.wk.d", "oic.d.bridge"],
   "if": ["oic.if.r", "oic.if.baseline"],
   "p": {"bm": 3},
   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
 },
 {
   "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
   "href": "/oic/p",
   "rt": ["oic.wk.p"],
   "if": ["oic.if.r", "oic.if.baseline"],
   "p": {"bm": 3},
   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
 },
 {
   "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
   "href": "/oic/sec/doxm",
   "rt": ["oic.r.doxm"],
   "if": ["oic.if.baseline"],
   "p": {"bm": 1},
   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
 },
 {
   "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
   "href": "/oic/sec/pstat",
   "rt": ["oic.r.pstat"],
   "if": ["oic.if.baseline"],
   "p": {"bm": 1},
   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
 },
 {
   "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
   "href": "/oic/sec/cred",
   "rt": ["oic.r.cred"],
   "if": ["oic.if.baseline"],
   "p": {"bm": 1},
   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
 },
 {
   "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
   "href": "/oic/sec/acl2",
   "rt": ["oic.r.acl2"],
   "if": ["oic.if.baseline"],
   "p": {"bm": 1},
   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
```

```
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/myIntrospection",
    "rt": ["oic.wk.introspection"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
  }
]
```

Response from the Fan VOD:

```
[
  {
    "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
    "href": "/oic/res",
    "rt": ["oic.wk.res"],
    "if": ["oic.if.ll", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
  },
  {
    "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
    "href": "/oic/d",
    "rt": ["oic.wk.d", "oic.d.fan", "oic.d.virtual"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
  },
  {
    "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
    "href": "/oic/p",
    "rt": ["oic.wk.p"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
  },
  {
    "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
    "href": "/myFan",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
  },
  {
    "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
    "href": "/oic/sec/doxm",
    "rt": ["oic.r.doxm"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
  },
  {
    "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
    "href": "/oic/sec/pstat",
    "rt": ["oic.r.pstat"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
  },
  {
```



```
    "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
    "href": "/oic/sec/cred",
    "rt": ["oic.r.cred"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
  },
  {
    "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
    "href": "/oic/sec/acl2",
    "rt": ["oic.r.acl2"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
  },
  {
    "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
    "href": "/myFanIntrospection",
    "rt": ["oic.wk.introspection"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
  }
]
```


Response from the first Light VOD:


```
[
  {
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "href": "/oic/res",
    "rt": ["oic.wk.res"],
    "if": ["oic.if.ll", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
  },
  {
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "href": "/oic/d",
    "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
  },
  {
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "href": "/oic/p",
    "rt": ["oic.wk.p"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
  },
  {
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "href": "/myLight",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
  },
  {
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "href": "/oic/sec/doxm",
```



```
    "rt": ["oic.r.doxm"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
  },
  {
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "href": "/oic/sec/pstat",
    "rt": ["oic.r.pstat"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
  },
  {
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "href": "/oic/sec/cred",
    "rt": ["oic.r.cred"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
  },
  {
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "href": "/oic/sec/acl2",
    "rt": ["oic.r.acl2"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
  },
  {
    "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
    "href": "/myLightIntrospection",
    "rt": ["oic.wk.introspection"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
  }
]
```


Response from the second Light VOD:


```
[
  {
    "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
    "href": "/oic/res",
    "rt": ["oic.wk.res"],
    "if": ["oic.if.ll", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
  },
  {
    "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
    "href": "/oic/d",
    "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
  },
  {
    "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
    "href": "/oic/p",
    "rt": ["oic.wk.p"],
    "if": ["oic.if.r", "oic.if.baseline"],
```



```
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
  },
  {
    "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
    "href": "/myLight",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
  },
  {
    "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
    "href": "/oic/sec/doxm",
    "rt": ["oic.r.doxm"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
  },
  {
    "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
    "href": "/oic/sec/pstat",
    "rt": ["oic.r.pstat"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
  },
   {
    "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
    "href": "/oic/sec/cred",
    "rt": ["oic.r.cred"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
  },
  {
    "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
    "href": "/oic/sec/acl2",
    "rt": ["oic.r.acl2"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
  },
  {
    "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
    "href": "/myLightIntrospection",
    "rt": ["oic.wk.introspection"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
  }
]
```


Figure 5 – /oic/res example responses

5.5 "Deep translation" vs. "on-the-fly"

When translating a service between a Bridged Protocol (e.g., AllJoyn) and OCF protocols, there are two possible types of translation. Bridge Platforms are expected to dedicate most of their logic to "deep translation" types of communication, in which data models used with the Bridged Protocol are mapped to the equivalent OCF Resource Types and vice-versa, in such a way that a compliant

OCF Client or Bridged Client would be able to interact with the service without realising that a translation was made.

"Deep translation" is out of the scope of this document, as the procedure far exceeds mapping of types. For example, clients on one side of a Bridge Platform may decide to represent an intensity as an 8-bit value between 0 and 255, whereas the devices on the other may have chosen to represent that as a floating-point number between 0.0 and 1.0. It's also possible that the procedure may require storing state in the Bridge Platform. Either way, the programming of such translation will require dedicated effort and study of the mechanisms on both sides.

The other type of translation, the "on-the-fly" or "one-to-one" translation, requires no prior knowledge of the device-specific schema in question on the part of the Bridge Platform. The burden is, instead, on one of the other participants in the communication, usually the client application. That stems from the fact that "on-the-fly" translation always produces Bridged Resource Types and OCF Resource Types as vendor extensions.

For AllJoyn, deep translation is specified in ISO/IEC 30118-6:2018, and on-the-fly translation is covered in clause 7.2 of this document.

5.6 Security

Please refer to ISO/IEC 30118-2:2018 for security specific requirements as they pertain to a Bridge Platform. These security requirements include both universal requirements applicable to all Bridged Protocols, and additional security requirements specific to each Bridged Protocol.

6 AllJoyn translation

6.1 Operational scenarios

The overall goals are to:

1) make Bridged Servers appear to OCF clients as if they were native OCF servers, and

2) make OCF servers appear to Bridged Clients as if they were native non-OCF servers.

6.2 Requirements specific to an AllJoyn Bridging Function

6.2.1 Introduction

The Bridge Platform shall be an AllJoyn Router Node. (This is a requirement so that users can expect that a certified Bridge will be able to talk to any AllJoyn device, without the user having to buy some other device.)

The requirements in clause 6.2 apply when using algorithmic translation, and by default apply to deep translation unless the relevant clause for such deep translation specifies otherwise.

6.2.2 Use of introspection

Whenever possible, the translation code should make use of metadata available that indicates what the sender and recipient of the message in question are expecting. For example, devices that are AllJoyn Certified are required to carry the introspection data for each object and interface they expose. When the metadata is available, Bridging Functions should convert the incoming payload to exactly the format expected by the recipient and should use information when translating replies to form a more useful message.

For example, for an AllJoyn specific Bridging Function, the expected interaction list is presented in Table 1.

Table 1 – AllJoyn Bridging Function Interaction List

Message Type	Sender	Receiver	Metadata
Request	AllJoyn 16.10	OCF 1.0	Available
Request	OCF 1.0	AllJoyn 16.10	Available
Response	AllJoyn 16.10	OCF 1.0	Available
Response	OCF 1.0	AllJoyn 16.10	Available

6.2.3 Stability and loss of data

Round-tripping through the translation process specified in this document is not expected to reproduce the same original message. The process is, however, designed not to lose data or precision in messages, though it should be noted that both OCF and AllJoyn payload formats allow for future extensions not considered in this document.

However, a third round of translation should produce the same identical message as was previously produced, provided the same information is available. That is, in the chain shown in , payloads 2 and 4 as well as 3 and 5 should be identical.

Figure 6 – Payload Chain.

6.2.4 Exposing AllJoyn producer devices to OCF clients

6.2.4.1 Virtual OCF Devices and Resources

As specified in ISO/IEC 30118-2:2018 the value of the "di" property of OCF Devices (including VODs) shall be established as part of Onboarding of that VOD.

Each AllJoyn object shall be mapped to one or more Virtual OCF Resources. If all AllJoyn interfaces can be translated to resource types on the same resource, there should be a single Virtual OCF Resource, and the path component of the URI of the Virtual OCF Resource shall be the AllJoyn object path, where each "_h" in the AllJoyn object path is transformed to "-" (hyphen), each "_d" in the AllJoyn object path is transformed to "." (dot), each "_t" in the AllJoyn object path is transformed to "~" (tilde), and each "_u" in the AllJoyn object path is transformed to "_" (underscore). Otherwise, a Resource with that path shall exist with a Resource Type of ["oic.wk.col", "oic.r.alljoynobject"] which is a Collection of links, where "oic.r.alljoynobject" is defined in clause 9.2 and the items in the collection are the Resources with the translated Resource Types.

The value of the "piid" property of "/oic/d" for each VOD shall be the value of the OCF-defined AllJoyn field "org.openconnectivity.piid" in the AllJoyn About Announce signal, if that field exists, else it shall be calculated by the Bridging Function as follows:

- – If the AllJoyn device supports security, the value of the "piid" property value shall be the peer GUID.
- – If the AllJoyn device does not support security but the device is being bridged anyway (see 9.2), the "piid" property value shall be derived from the DeviceId and AppId properties (in the About data), by concatenating the DeviceId value (not including any null termination) and the AppId bytes and using the result as the "name" to be used in the algorithm specified in IETF RFC 4122 clause 4.3, with SHA-1 as the hash algorithm, and 8f0e4e90-79e5-11e6-bdf4-0800200c9a66 as the name space ID. (This is to address the problem of being able to de-duplicate AllJoyn devices exposed via separate OCF Bridge Devices.)

A Bridging Function implementation is encouraged to listen for AllJoyn About Announce signals matching any AllJoyn interface name. It can maintain a cache of information it received from these signals, and use the cache to quickly handle "/oic/res" queries from OCF Clients (without having to wait for Announce signals while handling the queries).

A Bridging Function implementation is encouraged to listen for other signals (including EmitsChangedSignal of properties) only when there is a client subscribed to a corresponding resource on a Virtual AllJoyn Device.

There are multiple types of AllJoyn interfaces, which shall be handled as follows.

1) If the AllJoyn interface is in a well-defined set (defined in ISO/IEC 30118-6:2018 or 6.2.4.2) of interfaces where standard forms exist on both the AllJoyn and OCF sides, the Bridging Function shall either:
   a) follow the specification for translating that interface specially, or
   b) not translate the AllJoyn interface.
2) If the AllJoyn interface is not in the well-defined set, the Bridging Function shall either:
   a) not translate the AllJoyn interface, or
   b) algorithmically map the AllJoyn interface as specified in 6.3 to custom/vendor-defined Resource Types by converting the AllJoyn interface name to OCF resource type name(s).

An AllJoyn interface name shall be converted to a Device Type or a set of one or more OCF Resource Types as follows:

1) If the AllJoyn interface has any members, append a suffix ".<seeBelow>" where <seeBelow> is described in this clause.
2) For each upper-case letter present in the entire string, replace it with a hyphen followed by the lower-case version of that letter (e.g., convert "A" to "-a").
3) If an underscore appears followed by a (lower-case) letter or a hyphen, for each such occurrence, replace the underscore with two hyphens (e.g., convert "_a" to "--a", "_-a" to "---a").
4) For each underscore remaining, replace it with a hyphen (e.g., convert "_1" to "-1").
5) Prepend the "x." prefix.

Some examples are shown in Table 2. The first three are normal AllJoyn names converted to unusual OCF names. The last three are unusual AllJoyn names converted (perhaps back) to normal OCF names. ("xn--" is a normal domain name prefix for the Punycode-encoded form of an Internationalized Domain Name, and hence can appear in a normal vendor-specific OCF name.)

Table 2 – AllJoyn to OCF Name Examples

From AllJoyn name	To OCF name
example.Widget	x.example.-widget
example.my__widget	x.example.my----widget
example.My_Widget	x.example.-my---widget
xn_p1ai.example	x.xn--p1ai.example
xn__90ae.example	x.xn--90ae.example
example.myName_1	x.example.my-name-1

Each AllJoyn interface that has members and is using algorithmic mapping shall be mapped to one or more Resource Types as follows:

– AllJoyn Properties with the same EmitsChangedSignal value are mapped to the same Resource Type where the value of the <seeBelow> label is the value of EmitsChangedSignal. AllJoyn Properties with EmitsChangedSignal values of "const" or "false", are mapped to Resources that are not Observable, whereas AllJoyn Properties with EmitsChangedSignal values of "true" or "invalidates" result in Resources that are Observable. The Version property in an AllJoyn interface is always considered to have an EmitsChangedSignal value of "const", even if not specified in introspection XML. The name of each property on the Resource Type shall be "<ResourceType>.<AllJoynPropertyName>", where each "_d" in the <AllJoynPropertyName> is transformed to "." (dot), and each "_h" in the <AllJoynPropertyName> is transformed to "-" (hyphen).

– Resource Types mapping AllJoyn Properties with access "readwrite" shall support the "oic.if.rw" OCF Interface. Resource Types mapping AllJoyn Properties with access "read" shall support the "oic.if.r" OCF Interface. Resource Types supporting both the "oic.if.rw" and "oic.if.r" OCF Interfaces shall choose "oic.if.r" as the default Interface.

– Each AllJoyn Method is mapped to a separate Resource Type, where the value of the <seeBelow> label is the AllJoyn Method name. The Resource Type shall support the "oic.if.rw" OCF Interface. Each argument of the AllJoyn Method shall be mapped to a separate Property on the Resource Type, where the name of that Property is prefixed with "<ResourceType>arg<#>", where <#> is the 0-indexed position of the argument in the AllJoyn introspection xml, in order to help get uniqueness across all Resource Types on the same Resource. Therefore, when the AllJoyn argument name is not specified, the name of that property is "<ResourceType>arg<#>", where <#> is the 0-indexed position of the argument in the AllJoyn introspection XML. In addition, that Resource Type has an extra "<ResourceType>validity" property that indicates whether the rest of the properties have valid values. When the values are sent as part of an UPDATE response, the validity property is true, and any other properties have valid values. In a RETRIEVE (GET or equivalent in the relevant transport binding) response, the validity property is false, and any other properties can have meaningless values. If the validity property appears in an UPDATE request, its value shall be true (a value of false shall result in an error response).

– Each AllJoyn Signal (whether sessionless, sessioncast, or unicast) is mapped to a separate Resource Type on an Observable Resource, where the value of the <seeBelow> label is the AllJoyn Signal name. The Resource Type shall support the "oic.if.r" OCF Interface. Each argument of the AllJoyn Signal is mapped to a separate Property on the Resource Type, where the name of that Property is prefixed with "<ResourceType>arg<#>", where <#> is the 0-indexed position of the argument in the AllJoyn introspection xml, in order to help get uniqueness across all Resource Types on the same Resource. Therefore, when the AllJoyn argument name is not specified, the name of that property is "<ResourceType>arg<#>", where <#> is the 0-indexed position of the argument in the AllJoyn introspection XML. In addition, that Resource Type has an extra "<ResourceType>validity" property that indicates whether the rest of the properties have valid values. When the values are sent as part of a NOTIFY response, the validity property is true, and any other properties have valid values. In a RETRIEVE (GET or equivalent in the relevant transport binding) response, the validity property is false, and any other properties returned can have meaningless values. This is because in AllJoyn, the signals are instantaneous events, and the values are not necessarily meaningful beyond the lifetime of that message. Note that AllJoyn does have a TTL field that allows store-and-forward signals, but such support is not required in OCF 1.0. We expect that in the future, the TTL may be used to allow valid values in response to a RETRIEVE that is within the TTL.

When an algorithmic mapping is used, AllJoyn data types shall be mapped to OCF property types according to 6.3.

If an AllJoyn operation fails, the Bridging Function shall send an appropriate OCF error response to the OCF client. If an AllJoyn error name is available and does not contain the "org.openconnectivity.Error.Code" prefix, it shall construct an appropriate OCF error message (e.g., diagnostic payload if using CoAP) from the AllJoyn error name and AllJoyn error message (if any), using the form "<error name>: <error message>", with the <error name> taken from the AllJoyn error name field and the <error message> taken from the AllJoyn error message, and the CoAP error code set to an appropriate value (if CoAP is used). If an AllJoyn error name is available and contains the "org.openconnectivity.Error.Code" prefix, the OCF error message (e.g., diagnostic payload if using CoAP) should be taken from the AllJoyn error message (if any), and the CoAP error code (if CoAP is used) set to a value derived as follows; remove the "org.openconnectivity.Error.Code" prefix, and if the resulting error name is of the form "<#>" where <#> is an error code without a decimal (e.g., "404"), the CoAP error code shall be the error code indicated by the "<#>". Example: "org.openconnectivity.Error.Code404" becomes "404", which shall result in an error 4.04 for a CoAP transport.

6.2.4.2 Exposing an AllJoyn producer application as a Virtual OCF Server

Table 3 shows how OCF Device properties, as specified in Table 27 in ISO/IEC 30118-1:2018 shall be derived, typically from fields specified in the AllJoyn About Interface Specification and AllJoyn Configuration Interface Specification.

If the AllJoyn About or Config data field has a mapping rule defined (as in Table 3, Table 4, Table 5, and Table 6), the field name shall be translated based on that mapping rule; else if the AllJoyn About or Config data field has a fully qualified name (with a <domain> prefix (such as "com.example", "org.alljoyn"), the field name shall be translated based on the rules specified in 6.2.4 for mapping AllJoyn fields; else, the field shall not be translated as it may be incorrect (error) or it has no valid mapping (such as daemonRealm and passCode).

Table 3 – oic.wk.d resource type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From AJ Field name	AJ Description	AJ Mandatory
(Device) Name	n	Human friendly name For example, "Bob's Thermostat"	Y	AppName (no exact equivalent exists)	Application name assigned by the app manufacturer (developer or the OEM).	Y
Spec Version	icv	Spec version of ISO/IEC 30118-1:2018 this device is implemented to, the syntax is "core.major.minor"]	Y	(none)	Bridge Platform should return its own value	N
Device ID	di	Unique identifier for Device. This value shall be as defined in ISO/IEC 30118-2:2018 for DeviceID.	Y	(none)	Use as defined in ISO/IEC 30118-2:2018	N
Protocol-Independent ID	piid	Unique identifier for OCF Device (UUID)	Y	org.openconnectivity.piid if it exists, else "Peer GUID" (not in About, but exposed by protocol) if authenticated, else Hash(DeviceId,AppId) where the Hash is done by concatenating the Device Id (not	Peer GUID: The peer GUID is the only persistent identity for a peer. Peer GUIDs are used by the authentication mechanisms to uniquely identify a remote application instance. The peer	Per GUID: conditionally Y DeviceId: Y AppId: Y

				including any null terminator) and the AppId and using the algorithm in IETF RFC 4122 clause 4.3, with SHA-1. This means that the value of di may change if the resource is read both before and after authentication, in order to mitigate privacy concerns discussed in RFC 6973.	GUID for a remote peer is only available if the remote peer has been authenticated. DeviceId: Device identifier set by platform-specific means. AppId: A 128-bit globally unique identifier for the application. The AppId shall be a universally unique identifier as specified in IETF RFC 4122.	
Data Model Version	dmv	Spec version(s) of the vertical specifications this device data model is implemented to. The syntax is a comma separated list of "<vertical>.major.minor"]. <vertical> is the name of the vertical (i.e. sh for Smart Home)	Y	Comma separated list of the Version property values of each interface listed in the objectDescription argument of the Announce signal of About. In addition to the mandatory values specified in ISO/IEC 30118-1:2018, additional values are formatted as "x.<interface name>.<Version property value>".	This document assumes that the value of the Version property is the same as the value of the "org.gtk.GDBus.Since" annotation of the interface in the AllJoyn introspection XML, and therefore the value of the Version property may be determined through introspection alone. Note that AllJoyn specifies that the default value is 1 if the "org.gtk.GDBus.Since" annotation is absent.	N, but required by IRB for all standard interfaces, and absence can be used to imply a constant (e.g., 0)
Localized Descriptions	ld	Detailed description of the Device, in one or more languages. This property is an array of objects where each object has a 'language' field (containing an RFC 5646 language tag) and a 'value' field containing the device description in the indicated language.	N	Description	Detailed description expressed in language tags as in RFC 5646.	Y
Software Version	sv	Version of the device software.	N	SoftwareVersion	Software version of the app.	Y

Manufacturer Name	dmn	Name of manufacturer of the Device, in one or more languages. This property is an array of objects where each object has a ‘language’ field (containing an RFC 5646 language tag) and a ‘value’ field containing the manufacturer name in the indicated language.	N	Manufacturer	The manufacturer's name of the app.	Y
Model Number	dmno	Model number as designated by manufacturer.	N	ModelNumber	The app model number.	Y

In addition, any additional vendor-defined fields in the AllJoyn About data shall be mapped to vendor-defined properties in the OCF Device resource "/oic/d" (which implements the "oic.wk.d" resource type), with a property name formed by prepending "x." to the AllJoyn field name.

Table 4 shows how OCF Device Configuration properties, as specified in Table 22 in ISO/IEC 30118-1:2018 shall be derived:

Table 4 – oic.wk.con resource type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From AJ Field name	AJ Description	AJ Mandatory
(Device) Name	n	Human friendly name For example, "Bob’s Thermostat"	Y	AppName (no exact equivalent exists)	Application name assigned by the app manufacturer (developer or the OEM).	Y
Location	loc	Provides location information where available.	N	org.openconnectivity.loc (if it exists, else property shall be absent)		N
Location Name	locn	Human friendly name for location For example, "Living Room".	N	org.openconnectivity.locn (if it exists, else property shall be absent)		N
Currency	c	Indicates the currency that is used for any monetary transactions	N	org.openconnectivity.c (if it exists, else property shall be absent)		N
Region	r	Free form text Indicating the current region in which the device is located geographically. The free form	N	org.openconnectivity.r (if it exists, else property shall be absent)		N

		text shall not start with a quote (").				
Localized Names	ln	Human-friendly name of the Device, in one or more languages. This property is an array of objects where each object has a ‘language’ field (containing an RFC 5646 language tag) and a ‘value’ field containing the device name in the indicated language. If this property and the Device Name (n) property are both supported, the Device Name (n) value shall be included in this array.	N	AppName	Application name assigned by the app manufacturer (developer or the OEM).	Y
Default Language	dl	The default language supported by the Device, specified as an RFC 5646 language tag. By default, clients can treat any string property as being in this language unless the property specifies otherwise.	N	DefaultLanguage	The default language supported by the device. Specified as an IETF language tag listed in RFC 5646.	Y

In addition, any additional vendor-defined fields in the AllJoyn Configuration data shall be mapped to vendor-defined properties in the OCF Configuration resource (which implements the "oic.wk.con" resource type and optionally the "oic.wk.con.p" resource type), with a property name formed by prepending "x." to the AllJoyn field name.

Table 5 shows how OCF Platform properties, as specified in Table 28 in ISO/IEC 30118-1:2018 shall be derived, typically from fields specified in the AllJoyn About Interface Specification and AllJoyn Configuration Interface Specification.

935 Table 5 – oic.wk.p resource type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From AJ Field name	AJ Description	AJ Mandatory
Platform ID	pi	Unique identifier for the physical platform (UIUID); this shall be a UUID in accordance with IETF RFC 4122. It is recommended that the UUID be created using the random generation scheme (version 4 UUID) specific in the RFC.	Y	DeviceId if it is a UUID, else generate a name-based UUID from the DeviceId using the DeviceId value (not including any null termination) as the "name" to be used in the algorithm specified in IETF RFC 4122 clause 4.3, with SHA-1 as the hash algorithm, and 8f0e4e90-79e5-11e6-bdf4-0800200c9a66 as the name space ID.	Name of the device set by platform-specific means (such as Linux and Android).	Y
Manufacturer Name	mnmn	Name of manufacturer (not to exceed 16 characters)	Y	Manufacturer (in DefaultLanguage, truncated to 16 characters)	The manufacturer's name of the app.	Y
Manufacturer Details Link (URL)	mnml	URL to manufacturer (not to exceed 32 characters)	N	org.openconnectivity.mnml (if it exists, else property shall be absent)		N
Model Number	mnmo	Model number as designated by manufacturer	N	ModelNumber	The app model number.	Y
Date of Manufacture	mndt	Manufacturing date of device	N	DateOfManufacture	Date of manufacture using format YYYY-MM-DD (known as XML DateTime format).	N
Platform Version	mnpv	Version of platform – string (defined by manufacturer)	N	org.openconnectivity.mnpv (if it exists, else property shall be absent)		N
OS Version	mnos	Version of platform resident OS – string (defined by manufacturer)	N	org.openconnectivity.mnos (if it exists, else property shall be absent)		N
Hardware Version	mnhw	Version of platform hardware	N	HardwareVersion	Hardware version of the device on which	N

					the app is running.	
Firmware version	mnfv	Version of device firmware	N	org.openconnectivity.mnfv (if it exists, else property shall be absent)		N
Support URL	mnsl	URL that points to support information from manufacturer	N	SupportUrl	Support URL (populated by the manufacturer)	N
SystemTime	st	Reference time for the device	N	org.openconnectivity.st (if it exists, else property shall be absent)		N
Vendor ID	vid	Vendor defined string for the platform. The string is freeform and up to the vendor on what text to populate it.	N	DeviceId	Name of the device set by platform-specific means (such as Linux and Android).	Y

Table 6 shows how OCF Platform Configuration properties, as specified in Table 23 in the ISO/IEC 30118-1:2018 shall be derived:

Table 6 – oic.wk.con.p resource type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From AJ Field name	AJ Description	AJ Mandatory
Platform Names	Mnpn	Platform Identifier	N	DeviceName	Name of the device set by platform-specific means (such as Linux and Android).	Device name assigned by the user. The device name appears on the UI as the friendly name of the device.

In addition, the "oic.wk.mnt" properties Factory_Reset ("fr") and Reboot ("rb") shall be mapped to AllJoyn Configuration methods FactoryReset and Restart, respectively.

6.2.5 Exposing OCF resources to AllJoyn consumer applications

6.2.5.1 Use of AllJoyn Producer Application

Unless specified otherwise, each OCF resource shall be mapped to a separate AllJoyn object.

Each OCF Server shall be exposed as a separate AllJoyn producer application, with its own About data. This allows platform-specific, device-specific, and resource-specific fields to all be preserved across translation. However, this requires that AllJoyn Claiming of such producer applications be solved in a way that does not require user interaction, but this is left as an implementation issue.

The AllJoyn producer application shall implement the "oic.d.virtual" AllJoyn interface. This allows Bridge Platforms to determine if a device is already being translated when multiple Bridge Platforms are present. The "oic.d.virtual" interface is defined as follows:

```
<interface name="oic.d.virtual"/>
```

The implementation may choose to implement this interface by the AllJoyn object at path "/oic/d".

The AllJoyn peer ID shall be the OCF device ID ("di").

Unless specified otherwise, the AllJoyn object path shall be the OCF URI path, where each "-" (hyphen) in the OCF URI path is transformed to "_h", each "." (dot) in the OCF URI path is transformed to "_d", each "~" (tilde) in the OCF URI path is transformed to "_t", and each "_" (underscore) in the OCF URI path is transformed to "_u".

The AllJoyn About data shall be populated per Table 8.

A Bridging Function implementation is encouraged to maintain a cache of OCF resources to handle the implementation of queries from the AllJoyn side, and emit an Announce Signal for each OCF Server. Specifically, the implementation could always Observe "/oic/res" changes and only Observe other resources when there is a client with a session on a Virtual AllJoyn Device.

There are multiple types of resources, which shall be handled as follows.

1) If the Resource Type is in a well-defined set (defined in ISO/IEC 30118-6:2018 or 6.2.5.2) of resource types where standard forms exist on both the AllJoyn and OCF sides, the Bridging Function shall either:
   a) follow the specification for translating that resource type specially, or
   b) not translate the Resource Type.
2) If the Resource Type is not in the well-defined set (but is not a Device Type), the Bridging Function shall either:
   a) not translate the Resource Type, or
   b) algorithmically map the Resource Type as specified in 6.3 to a custom/vendor-defined AllJoyn interface by converting the OCF Resource Type name to an AllJoyn Interface name.

An OCF Resource Type or Device Type shall be converted to an AllJoyn interface name as follows:

1) Remove the "x." prefix if present
2) For each occurrence of a hyphen (in order from left to right in the string):
   a) If the hyphen is followed by a letter, replace both characters with a single upper-case version of that letter (e.g., convert "-a" to "A").
   b) Else, if the hyphen is followed by another hyphen followed by either a letter or a hyphen, replace two hyphens with a single underscore (e.g., convert "--a" to "_a", "---" to "_-").
   c) Else, convert the hyphen to an underscore (i.e., convert "-" to "_").

Some examples are shown in the Table 7. The first three are unusual OCF names converted (perhaps back) to normal AllJoyn names. The last three are normal OCF names converted to unusual AllJoyn names. ("xn--" is a normal domain name prefix for the Punycode-encoded form of an Internationalized Domain Name, and hence can appear in a normal vendor-specific OCF name.)

Table 7 – Example name mapping

From OCF name	To AllJoyn name
x.example.-widget	example.Widget
x.example.my----widget	example.my__widget
x.example.-my---widget	example.My_Widget
x.xn--p1ai.example	xn_p1ai.example
x.xn--90ae.example	xn__90ae.example
x.example.my-name-1	example.myName_1

An OCF Device Type is mapped to an AllJoyn interface with no members.

Unless specified otherwise, each OCF Resource Type shall be mapped to an AllJoyn interface as follows:

– Each OCF property is mapped to an AllJoyn property in that interface, where each "." (dot) in the OCF property is transformed to "_d", and each "-" (hyphen) in the OCF property is transformed to "_h".

– The EmitsChangedSignal value for each AllJoyn property shall be set to "true" if the resource supports NOTIFY, or "false" if it does not. (The value is never set to "const" or "invalidates" since those concepts cannot currently be expressed in OCF.)

– The "access" attribute for each AllJoyn property shall be "read" if the OCF property is read-only, or "readwrite" if the OCF property is read-write.

– If the resource supports DELETE, a Delete() method shall appear in the interface.

– If the resource supports CREATE, a Create() method shall appear in the interface, with input arguments of each property of the resource to create. (Such information is not available algorithmically can be determined via introspection.) If such information is not available, a CreateWithDefaultValues() method shall appear which takes no input arguments. In either case, the output argument shall be an OBJECT_PATH containing the path of the created resource.

– If the resource supports UPDATE (i.e., the "oic.if.rw" or "oic.if.a" OCF Interface) then an AllJoyn property set operation (i.e., an org.freedesktop.DBus.Properties.Set() method call) shall be mapped to a Partial UPDATE (e.g., POST in CoAP) with the corresponding OCF property.

– If a Resource has a Resource Type "oic.r.alljoynobject", then instead of separately translating each of the Resources in the collection to its own AllJoyn object, all Resources in the collection shall instead be translated to a single AllJoyn object whose object path is the OCF URI path of the collection.

OCF property types shall be mapped to AllJoyn data types according to 6.3.

If an OCF operation fails, the Bridging Function shall send an appropriate AllJoyn error response to the AllJoyn consumer. If an error message is present in the OCF response, and the error message (e.g., diagnostic payload if using CoAP) fits the pattern "<error name>: <error message>" where <error name> conforms to the AllJoyn error name syntax requirements, the AllJoyn error name and AllJoyn error message shall be extracted from the error message in the OCF response. Otherwise, the AllJoyn error name shall be "org.openconnectivity.Error.Code<#>" where <#> is the error code (e.g., CoAP error code) in the OCF response without a decimal (e.g., "404") and the AllJoyn error message is the error message in the OCF response.

6.2.5.2 Exposing an OCF server as a Virtual AllJoyn Producer

The object description returned in the About interface shall be formed as specified in the AllJoyn About Interface Specification, and Table 8 shows how AllJoyn About Interface fields shall be derived, based on properties in "oic.wk.d", "oic.wk.con", "oic.wk.p", and "oic.wk.con.p".

Table 8 – AllJoyn about data fields

To AJ Field name	AJ Description	AJ Mandatory	From OCF Property title	OCF Property name	OCF Description	OCF Mandatory
AppId	A 128-bit globally unique identifier for the application. The AppId shall be a universally unique identifier as specified in RFC 4122.	Y	Device ID (no exact equivalent exists)	di	Unique identifier for OCF Device (UUID)	Y
DefaultLanguage	The default language supported by the device. Specified as an IETF language tag listed in RFC 5646.	Y	Default Language	dl	The default language supported by the Device, specified as an RFC 5646 language tag. By default, clients can treat any string property as being in this language unless the property specifies otherwise. If absent, the Bridge Platform shall return a constant, e.g., empty string	N
DeviceName (per supported language)	Name of the device set by platform-specific means (such as Linux and Android).	N	Platform Names	mnpn	Friendly name of the Platform. This property is an array of objects where each object has a ‘language’ field (containing an RFC 5646 language tag) and a ‘value’ field containing the platform friendly name in the indicated language. For example,	N

					[{"language":"en", "value":"Dave's Laptop"}]	
DeviceId	Device identifier set by platform-specific means.	Y	Platform ID	pi	Platform Identifier	Y
AppName (per supported language)	Application name assigned by the app manufacturer (developer or the OEM).	Y	Localized Names, if it exists, else (Device) Name	In or n	Human-friendly name of the Device, in one or more languages. This property is an array of objects where each object has a 'language' field (containing an RFC 5646 language tag) and a 'value' field containing the device name in the indicated language. If this property and the Device Name (n) property are both supported, the Device Name (n) value shall be included in this array.	N (In), Y (n)
Manufacturer (per supported language)	The manufacturer's name of the app.	Y	Manufacturer Name	dmn	Name of manufacturer of the Device, in one or more languages. This property is an array of objects where each object has a 'language' field (containing an RFC 5646 language tag) and a 'value' field containing the manufacturer name in the indicated language.	N
ModelNumber	The app model number.	Y	Model Number	dmno	Model number as designated by manufacturer	N
SupportedLanguages	List of supported languages.	Y	language fields of Localized Names	In	If In is supported, return the list of values of the language field of each array element, else return empty array	N

Description (per supported language)	Detailed description expressed in language tags as in RFC 5646.	Y	Localized Descriptions	ld	Detailed description of the Device, in one or more languages. This property is an array of objects where each object has a ‘language’ field (containing an RFC 5646 language tag) and a ‘value’ field containing the device description in the indicated language.	N
DateOfManufacture	Date of manufacture using format YYYY-MM-DD (known as XML DateTime format).	N	Date of Manufacture	mndt	Manufacturing date of device	N
SoftwareVersion	Software version of the app.	Y	Software Version	sv	Software version of the device.	N
AJSoftwareVersion	Current version of the AllJoyn SDK used by the application.	Y	(none)		Bridge Platform should return its own value	
HardwareVersion	Hardware version of the device on which the app is running.	N	Hardware Version	mnhw	Version of platform hardware	N
SupportUrl	Support URL (populated by the manufacturer).	N	Support URL	mnsl	URL that points to support information from manufacturer	N
org.openconnectivity.mnml		N	Manufacturer Details Link (URL)	mnml (if it exists, else field shall be absent)	URL to manufacturer (not to exceed 32 characters)	N
org.openconnectivity.mnpv		N	Platform Version	mnpv (if it exists, else field shall be absent)	Version of platform – string (defined by manufacturer)	N
org.openconnectivity.mnos		N	OS Version	mnos (if it exists, else field	Version of platform resident OS – string (defined by manufacturer)	N

				shall be absent)		
org.openconnectivity.mnfv		N	Firmware version	mnfv (if it exists, else field shall be absent)	Version of device firmware	N
org.openconnectivity.st		N	SystemTime	st (if it exists, else field shall be absent)	Reference time for the device	N
org.openconnectivity.piid		N	Protocol-Independent ID	piid	A unique and immutable Device identifier. A Client can detect that a single Device supports multiple communication protocols if it discovers that the Device uses a single Protocol Independent ID value for all the protocols it supports.	Y

The AllJoyn field "org.openconnectivity.piid" shall be announced but shall not be localized and its D-Bus type signature shall be "s". All other AllJoyn field names listed in Table 5 which have the prefix "org.openconnectivity." shall be neither announced nor localized and their D-Bus type signature shall be "s".

In addition, any additional vendor-defined properties in the OCF Device resource "/oic/d" (which implements the "oic.wk.d" resource type) and the OCF Platform resource "/oic/p" (which implements the "oic.wk.p" resource type) shall be mapped to vendor-defined fields in the AllJoyn About data, with a field name formed by removing the leading "x." from the property name.

Table 9 shows how AllJoyn Configuration Interface fields shall be derived, based on properties in "oic.wk.con" and "oic.wk.con.p".

Table 9 – AllJoyn configuration data fields

To AJ Field name	AJ Description	AJ Mandatory	From OCF Property title	OCF Property name	OCF Description	OCF Mandatory
DefaultLanguage	Default language supported by the device.	N	Default Language	dl	The default language supported by the Device, specified as an RFC 5646 language tag. By default, clients can treat any string property as being in this	N

					language unless the property specifies otherwise.	
DeviceName	Device name assigned by the user. The device name appears on the UI as the friendly name of the device.	N	PlatformNames	mnpn	Friendly name of the Platform. This property is an array of objects where each object has a ‘language’ field (containing an RFC 5646 language tag) and a ‘value’ field containing the platform friendly name in the indicated language. For example, [{"language":"en", "value":"Dave’s Laptop"}]	N
org.openconnectivity.loc		N	Location	loc (if it exists, else field shall be absent)	Provides location information where available.	N
org.openconnectivity.locn		N	Location Name	locn (if it exists, else field shall be absent)	Human friendly name for location For example, "Living Room".	N
org.openconnectivity.c		N	Currency	c (if it exists, else field shall be absent)	Indicates the currency that is used for any monetary transactions	N
org.openconnectivity.r		N	Region	r (if it exists, else field shall be absent)	Free form text Indicating the current region in which the device is located geographically. The free form text shall not start with a quote (").	N

The AllJoyn field "org.openconnectivity.loc" shall be neither announced nor localized and its D-Bus type signature shall be "ad". All other AllJoyn field names listed in Table 5 which have the prefix "org.openconnectivity." shall be neither announced nor localized and their D-Bus type signature shall be "s".

In addition, the Configuration methods FactoryReset and Restart shall be mapped to "oic.wk.mnt" properties Factory_Reset ("fr") and Reboot ("rb"), respectively, and any additional vendor-defined properties in the OCF Configuration resource (which implements the "oic.wk.con" resource type and optionally the "oic.wk.con.p" resource type) shall be mapped to vendor-defined fields the AllJoyn Configuration data, with a field name formed by removing the leading "x." from the property name.

6.2.6 Security

For AllJoyn bridging, an OCF Onboarding Tool shall be able to block the communication of all OCF Devices with all Bridged Devices that don’t communicate securely with the Bridge, by using the Bridge Device’s "oic.r.securemode" Resource.

6.3 On-the-Fly Translation from D-Bus and OCF payloads

6.3.1 Introduction

The "dbus1" payload format is specified in the D-Bus Specification and AllJoyn adopted the D-Bus protocol and made it distributed over the network. The modifications done by AllJoyn to the format are all in the header part of the packet, not in the data payload itself, which remains compatible with "dbus1". Other variants of the protocol that have been proposed by the Linux community ("GVariant" and "kdbus" payloads) contain slight incompatibilities and are not relevant for this discussion.

6.3.2 Translation without aid of introspection

6.3.2.1 Introduction

Clause 6.3.2 describes how Bridging Functions shall translate messages between the two payload formats in the absence of introspection metadata from the actual device. This situation arises in the when there is content not described by introspection, such as the inner payload of AllJoyn properties of type "D-Bus VARIANT".

Since introspection is not available, the Bridging Function cannot know the rich JSON sub-type, only the underlying CBOR type and from that it can infer the JSON generic type, and hence translation is specified in terms of those generic types.

6.3.2.2 Booleans

Boolean conversion is trivial since both sides support this type.

Table 10 – Boolean translation

D-Bus type	JSON type
"b" – BOOLEAN	boolean (true or false)

6.3.2.3 Numeric types

The translation of numeric types is lossy and that is unavoidable due to the limited expressiveness of the JSON generic types. This can only be solved with introspection.

The translation of numeric types is direction-specific.

Table 11 – Numeric type translation, D-Bus to JSON

From D-Bus type	To JSON type
"y" - BYTE (unsigned 8-bit)	Number

"n" - UINT16 (unsigned 16-bit)	
"u" - UINT32 (unsigned 32-bit)	
"t" - UINT64 (unsigned 64-bit)[a]	
"q" - INT16 (signed 16-bit)	
""" - INT32 (signed 32-bit)	
"x" - INT64 (signed 64-bit)[a]	
"d" - DOUBLE (IEEE 754 double precision)	
[a] D-Bus payloads of types "t" (UINT64) and "x" (INT64) can contain values that cannot be perfectly represented in IEEE 754 double-precision floating point. The RFCs governing JSON do not forbid such numbers but caution that many implementations may not be able to deal with them. Currently, OCF transports its payload using CBOR instead of JSON, which can represent those numbers with fidelity. However, it should be noted that ISO/IEC 30118-1:2018 does not allow for integral numbers outside the range -2^53≤x≤2^53.	

Table 12 – Numeric type translation, JSON to D-Bus

From JSON type	To D-Bus type
number	"d" - DOUBLE[a]
[a] To provide the most predictable result, all translations from OCF to AllJoyn produce values of type "d" DOUBLE (IEEE 754 double precision).	

6.3.2.4 Text strings

Table 13 – Text string translation

D-Bus type	JSON type
"s" – STRING	string

Conversion between D-Bus and JSON strings is simple, as both require their content to be valid Unicode. For example, an implementation can typically do a direct byte copy, as both protocols specify UTF-8 as the encoding of the data, neither constrains the data to a given normalisation format nor specify whether private-use characters or non-characters should be disallowed.

Since the length of D-Bus strings is always known, it is recommended Bridging Functions not use CBOR indeterminate text strings (first byte 0x7f).

6.3.2.5 Byte arrays

The translation of a byte array is direction-specific.

Table 14 – Byte array translation

From D-Bus type	To JSON type
"ay" - ARRAY of BYTE	(base64-encoded) string

The base64url encoding is specified in IETF RF 4648 clause 5.

6.3.2.6 D-Bus variants

Table 15 – D-Bus variant translation

D-Bus type	JSON type
"v" – VARIANT	see clause 6.3.2.6

D-Bus has a type called VARIANT ("v") that is a wrapper around any other D-Bus type. It's a way for the type system to perform type-erasure. JSON, on the other hand, is not type-safe, which means that all JSON values are, technically, variants. The conversion for a D-Bus variant to JSON is performed by entering that variant and encoding the type carried inside as per the rules in this document.

The algorithm must be recursive, as D-Bus variants are allowed to contain variants themselves.

6.3.2.7 D-Bus object paths and signatures

The translation of D-Bus object paths and signatures is unidirectional (there is no mapping *to* them, only *from* them). This is shown in Table 16. In the reverse direction, clause 6.3.2.4 always converts to D-Bus STRING rather than OBJECT_PATH or SIGNATURE since it is assumed that "s" is the most common string type in use.

Table 16 – D-Bus object path translation

From D-Bus type	To JSON type
"o" - OBJECT_PATH	string
"g" – SIGNATURE	

Both D-Bus object paths and D-Bus type signatures are US-ASCII strings with specific formation rules, found in the D-Bus Specification. They are very seldom used and are not expected to be found in resources subject to translation without the aid of introspection.

6.3.2.8 D-Bus structures

The translation of the types in Table 17 is direction-specific:

Table 17 – D-Bus structure translation

From D-Bus type	To JSON type
"r" – STRUCT	array, length > 0

D-Bus structures can be interpreted as a fixed-length array containing a pre-determined list of types for each member. This is how such a structure is mapped to JSON: as an array of heterogeneous content, which are the exact members of the D-Bus structure, in the order in which they appear in the structure.

6.3.2.9 Arrays

The translation of the types in Table 18 is bidirectional:

Table 18 – Byte array translation

D-Bus type	JSON type
"ay" - ARRAY of BYTE	(base64-encoded) string – see 6.3.2.5
"ae" - ARRAY of DICT_ENTRY	object – see 6.3.2.10

The translation of the types in Table 19 is direction-specific:

Table 19 – Other array translation

From D-Bus type	To JSON type
"a" – ARRAY of anything else not specified	array

Aside from arrays of bytes and arrays of dictionary entries, which are mapped to JSON strings and objects respectively, arrays in JSON cannot be constrained to a single type (i.e., heterogeneous arrays). For that reason, strictly speaking all D-Bus arrays excepting arrays of bytes and arrays of dictionary entries must first be converted to arrays of variant "av" and then that array can be converted to JSON. See Table 20.

Table 20 – JSON array translation

From JSON type	Condition	To D-Bus type
array	length=0	"av" – ARRAY of VARIANT
array	length>0, all elements of same type	"a" – ARRAY
array	length>0, elements of different types	"r" – STRUCT

Conversion of D-Bus arrays of variants uses the conversion of variants as specified, which simply eliminates the distinction between a variant containing a given value and that value outside a variant. In other words, the elements of a D-Bus array are extracted and sent as elements of the JSON array, as per the other rules of this document.

6.3.2.10 Dictionaries / Objects

The choice of "dictionary of STRING to VARIANT" is made because that is the most common type of dictionary found in payloads and is an almost perfect superset of all possible dictionaries in D-Bus anyway. Moreover, it can represent JSON Objects with fidelity, which is the representation that OCF uses in its data models, which in turn means those D-Bus dictionaries will be able to carry with fidelity any OCF JSON Object in current use. See Table 21

Table 21 – D-Bus dictionary translation

D-Bus type	JSON type
"a{sv}" - dictionary of STRING to VARIANT	object

D-Bus dictionaries that are not mapping string to variant are first converted to those constraints and then encoded in CBOR.

6.3.2.11 Non-translatable types

The types in Table 22 are not translatable, and the Bridging Function should drop the incoming message. None of the types in Table 22 are in current use by either AllJoyn or OCF 1.0 devices, so the inability to translate them should not be a problem.

Table 22 – Non-translation types

Type Scope	Type Name	Description
D-Bus	"h"	UNIX_FD (Unix File Descriptor)
JSON	Null	
JSON	undefined	Not officially valid JSON, but some implementations permit it

6.3.2.12 Examples

Table 23 and Table 24 provide some translation examples.

Table 23 – D-Bus to JSON translation examples

Source D-Bus	JSON Result
BOOLEAN(FALSE)	false
BOOLEAN(TRUE)	true
VARIANT(BOOLEAN(FALSE))	false
VARIANT(BOOLEAN(TRUE))	true
BYTE(0)	0.0
BYTE(255)	255.0
INT16(0)	0.0
INT16(-1)	-1.0
INT16(-32768)	-32768.0
UINT16(0)	0.0
UINT16(65535)	65535.0
INT32(0)	0.0
INT32(-2147483648)	-2147483648.0
INT32(2147483647)	2147483647.0
UINT32(0)	0.0
UINT32(4294967295)	4294967295.0
INT64(0)	0.0
INT64(-1)	-1.0
UINT64(18446744073709551615)	18446744073709551615.0(1)
DOUBLE(0.0)	0.0
DOUBLE(0.5)	0.5
STRING("")	""
STRING("Hello")	"Hello"
ARRAY<BYTE>()	""
ARRAY<BYTE>(0x48, 0x65, 0x6c, 0x6c, 0x6f)	"SGVsbG8"
OBJECT_PATH("/")	"/"
SIGNATURE()	""
SIGNATURE("s")	"s"
VARIANT(INT32(0))	0
VARIANT(VARIANT(INT32(0)))	0
VARIANT(STRING("Hello"))	"Hello"

Table 24 – JSON to D-Bus translation examples

Source JSON	D-Bus Result
false	BOOLEAN(false)
true	BOOLEAN(true)
0	DOUBLE(0.0)
-1	DOUBLE(-1.0)
-2147483648	DOUBLE(-2147483648.0)
2147483647	DOUBLE(2147483647.0)
2147483648	DOUBLE(2147483648.0)
-2147483649	DOUBLE(-2147483649.0)
9223372036854775808[(1)]	DOUBLE(9223372036854775808.0)
0.0	DOUBLE(0.0)
0.5	DOUBLE(0.5)
0.0f	DOUBLE(0.0)
0.5f	DOUBLE(0.5)
""	STRING("")
"Hello"	STRING("Hello")
[]	ARRAY<VARIANT>()
[1]	ARRAY<DOUBLE>(DOUBLE(1.0))
[1, 2147483648, false, "Hello"]	STRUCT<DOUBLE, DOUBLE, BOOLEAN, STRING>(DOUBLE(1.0), DOUBLE(2147483648.0), BOOLEAN(false), STRING("Hello"))
{}	map<STRING, VARIANT>()
{1: 1}	map<STRING, VARIANT>("1" → VARIANT(DOUBLE(1.0)))
{"1": 1}	map<STRING, VARIANT>("1" → VARIANT(DOUBLE(1.0)))

{"rep":
{
"state": false,
"power": 1.0,
"name": "My Light"
}
}
	map<STRING, VARIANT>(
	{STRING("rep"), VARIANT(map<STRING, VARIANT>(
	{STRING("state") → VARIANT(BOOLEAN(FALSE))},
	{STRING("power") → VARIANT(DOUBLE(1.0))},
	{STRING("name") → VARIANT(STRING("My Light"))}
	))}
	)

NOTE This value cannot be represented with IEEE754 double-precision floating point without loss of information. It is also outside the currently-allowed range of integrals in OCF.

6.3.3 Translation with aid of introspection

6.3.3.1 Introduction to Introspection Metadata

When introspection is available, the Bridging Function can use the extra metadata provided by the side offering the service to expose a higher-quality reply to the other side. This chapter details modifications to the translation described in the previous chapter when the metadata is found.

Introspection metadata can be used for both translating requests to services and replies from those services. When used to translate requests, the introspection is "constraining", since the Bridging Function must conform exactly to what that service expects. When used to translate replies, the introspection is "relaxing", but may be used to inform the receiver what other possible values may be encountered in the future.

Note that OCF introspection uses JSON types, media attributes, and format attributes, not CBOR encoding. The actual encoding of each JSON type is discussed in clause 12.4 of ISO/IEC 30118-1:2018 , JSON format attribute values are as defined in JSON Schema Validation, and JSON media attribute values are as defined in JSON Hyper-Schema.

6.3.3.2 Translation of the introspection itself

Note that both OCF 1.0 and AllJoyn require all services exposed to include introspection metadata, which means the Bridging Function will need to translate the introspection information on-the-fly for each OCF resource or AllJoyn producer it finds. The Bridging Function shall preserve as much of the original information as can be represented in the translated format. This includes both the information used in machine interactions and the information used in user interactions, such as description and documentation text.

6.3.3.3 Variability of introspection data

Introspection data is not a constant and the Bridging Function may find, upon discovering further services, that the D-Bus interface or OCF Resource Type it had previously encountered is different than previously seen. The Bridging Function needs to take care about how the destination side will react to a change in introspection.

D-Bus interfaces used by AllJoyn services may be updated to newer versions, which means a given type of service may be offered by two distinct versions of the same interface. Updates to standardised interfaces must follow strict guidelines established by the AllSeen Interface Review Board, mapping each version to a different OCF Resource Type should be possible without much difficulty. However, there’s no guarantee that vendor-specific extensions follow those requirements. Indeed, there’s nothing preventing two revisions of a product to contain completely incompatible interfaces that have the same name and version number.

On the opposite direction, the rules are much laxer. Since OCF specifies optional properties to its Resource Types, a simple monotonically-increasing version number like AllJoyn consumer applications expect is not possible.

However, it should be noted that services created by the Bridging Function by "on-the-fly" translation will only be accessed by generic client applications. Dedicated applications will only use "deep binding" translation.

6.3.3.4 Numeric types

For numeric values, all D-Bus and JSON numeric types are treated equally as source and may all be translated into any of the other side’s types. When translating a request to a service, the Bridging Function need only verify whether there would be loss of information when translating from source to destination. For example, when translating the number 1.5 to either a JSON integer or to one of the D-Bus integral types, there would be loss of information, in which case the Bridging Function should refuse the incoming message. Similarly, the value 1,234,567 does not fit the range of a D-Bus byte, 16-bit signed or unsigned integer.

When translating the reply from the service, the Bridging Function shall use the following rules.

Table 25 indicates how to translate from a JSON type to the corresponding D-Bus type, where the first matching row shall be used. If the JSON schema does not indicate the minimum value of a JSON integer, 0 is the default. If the JSON schema does not indicate the maximum value of a JSON integer, 2^{32} – 1 is the default. The resulting AllJoyn introspection XML shall contain "org.alljoyn.Bus.Type.Min" and "org.alljoyn.Bus.Type.Max" annotations whenever the minimum or maximum, respectively, of the JSON value is different from the natural minimum or maximum of the D-Bus type.

Table 25 – JSON type to D-Bus type translation

From JSON type	Condition	To D-Bus Type
integer	minimum ≥ 0 AND maximum < 2^8	"y" (BYTE)
	minimum ≥ 0 AND maximum < 2^{16}	"q" (UINT16)
	minimum ≥ -2^{15} AND maximum < 2^{15}	"n" (INT16)
	minimum ≥ 0 AND maximum < 2^{32}	"u" (UINT32)
	minimum ≥ -2^{31} AND maximum < 2^{31}	"i" (INT32)
	minimum ≥ 0	"t" (UINT64)
		"x" (INT64)
Number		"d" (DOUBLE)
String	pattern = "^0|([1-9][0-9]{0,19})$"	"t" (UINT64)
	pattern = "^0|(-?[1-9][0-9]{0,18)}$"	"x" (INT64)

Table 26 indicates how to translate from a D-Bus type to the corresponding JSON type.

Table 26 – D-Bus type to JSON type translation

From D-Bus type	To JSON type	Note
"y" (BYTE)	integer	"minimum" and "maximum" in the JSON schema shall be set to the value of the "org.alljoyn.Bus.Type.Min" and "org.alljoyn.Bus.Type.Max" (respectively) annotations if present, or to the min and max values of the D-Bus type's range if such annotations are absent.
"n" (UINT16)		
"q" (INT16)		
"u" (UINT32)		
"i" (INT32)		
"t" (UINT64)	integer if org.alljoyn.Bus.Type.Max ≤ 2^{53}, else string with JSON pattern attribute "^0|([1-9][0-9]{0,19})$".	IETF RFC 7159 clause 6 explains that higher JSON integers are not interoperable.
"x" (INT64)	integer (if org.alljoyn.Bus.Type.Min ≥ -2^{53} AND org.alljoyn.Bus.Type.Max ≤ 2^{53}), else string with JSON pattern attribute "^0|(-?[1-9][0-9]{0,18)}$".	IETF RFC 7159 clause 6 explains that other JSON integers are not interoperable.
"d" (double)	number	

6.3.3.5 Text string and byte arrays

There's no difference in the translation of text strings and byte arrays compared to clause 6.3.2. Clause 6.3.3 simply lists the JSON equivalent types for the generated OCF introspection. See Table 27.

Table 27 – Text string translation

D-Bus Type	JSON type	JSON media attribute, binaryEncoding property
"s" – STRING	string	(none)
"ay" - ARRAY of BYTE	string	base64

In addition, the mapping of the JSON Types in Table 28 is direction-specific:

Table 28 – JSON UUID string translation

From JSON type	Condition	To D-Bus Type
string	pattern = "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$"	"ay" – ARRAY of BYTE

JSON strings with any other format value (e.g., date-time, uri, etc.) or pattern value not shown in Table 28 shall be treated the same as if the format and pattern attributes were absent, by simply mapping the value to a D-Bus string.

6.3.3.6 D-Bus Variants

If the introspection of an AllJoyn producer indicates a value in a request should be a D-Bus VARIANT, the Bridging Function should create such a variant and encode the incoming value as the variant's payload as per the rules in the rest of this document. See Table 29.

Table 29 – D-Bus variant translation

D-Bus Type	JSON Type
"v" – VARIANT	see clause 6.3.3.6

6.3.3.7 D-Bus Object Paths and Signatures

If the introspection of an AllJoyn producer indicates a value in a request should be a D-Bus Object Path or D-Bus Signature, the Bridging Function should perform a validity check in the incoming CBOR Text String. If the incoming data fails to pass this check, the message should be rejected. See Table 30.

Table 30 – D-Bus object path translation

From D-Bus Type	To JSON Type
"o" – OBJECT_PATH	string
"g" – SIGNATURE	

6.3.3.8 D-Bus structures

D-Bus structure members are described in the introspection XML with the "org.alljoyn.Bus.Struct.*StructureName.*Field.*fieldName.*Type" annotation. The Bridging Function shall use the AJSoftwareVersion field of the About data obtained from a bridged AllJoyn producer as follows. When the version of AllJoyn implemented on the Bridged Device is v16.10.00 or greater and the member annotations are present, the Bridging Function shall use a JSON object to represent a structure, mapping each member to the entry with that name. The Bridging Function needs to be aware that the incoming CBOR payload may have changed the order of the fields, when compared to the D-Bus structure. When the version of AllJoyn implemented on the Bridged Device is less than v16.10.00, the Bridging Function shall follow the rule for translating D-Bus structures without the aid of introspection data.

6.3.3.9 Arrays and dictionaries

If the introspection of the AllJoyn interface indicates that the array is neither an ARRAY of BYTE ("ay") nor an ARRAY of VARIANT ("av") or that the dictionary is not mapping STRING to VARIANT ("a{sv}"), the Bridging Function shall apply the constraining or relaxing rules specified in other clauses.

Similarly, if the OCF introspection indicates a homogeneous array type, the information about the array's element type should be used as the D-Bus array type instead of VARIANT ("v").

6.3.3.10 Other JSON format attribute values

The JSON format attribute may include other custom attribute types. They are not known at this time, but it is expected that those types be handled by their type and representation alone.

6.3.3.11 Examples

Table 31 and Table 32 provide examples using introspection.

Table 31 – Mapping from AllJoyn using introspection

<table>
<tr><th>AllJoyn Source</th><th>AllJoyn Introspection Notes</th><th>Translated JSON Payload</th><th>OCF Introspection Notes</th></tr>
<tr><td>UINT32 (0)</td><td></td><td>0</td><td>JSON schema should indicate:
"type": "integer",
"minimum": 0,
"maximum": 4294967295</td></tr>
<tr><td>INT64 (0)</td><td></td><td>0</td><td>Since no Min/Max annotations exist in AllJoyn,
JSON schema should indicate:
"type": "string",
"pattern": "^0|(-?[1-9][0-9]{0,18})$"</td></tr>
<tr><td>UINT64 (0)</td><td></td><td>"0"</td><td>Since no Max annotation exists in AllJoyn, JSON schema should indicate:
"type": "string",
"pattern": ^0|([1-9][0-9]{0,19})$"</td></tr>
<tr><td>STRING("Hello")</td><td></td><td>"Hello"</td><td>JSON schema should indicate:
"type": "string"</td></tr>
<tr><td>OBJECT_PATH("/")</td><td></td><td>"/"</td><td>JSON schema should indicate:
"type": "string"</td></tr>
<tr><td>SIGNATURE("g")</td><td></td><td>"g"</td><td>JSON schema should indicate:
"type": "string"</td></tr>
<tr><td>ARRAY<BYTE>(0x48, 0x65, 0x6c, 0x6c, 0x6f)</td><td></td><td>"SGVsbG8"</td><td>JSON schema should indicate:
"type": "string",
"media binaryEncoding": "base64"</td></tr>
<tr><td>VARIANT(anything)</td><td></td><td>?</td><td>JSON schema should indicate:</td></tr>
</table>

			"type": ["boolean", "object", "array", "number", "string", "integer"]
ARRAY<INT32>()		[]	JSON schema should indicate: "type": "array", "items": { "type": "integer" }
ARRAY<INT64>()		[]	JSON schema should indicate: "type": "array", "items": { "type": "string", "pattern": "^0|([1-9][0-9]{0,18})$" }
STRUCT< INT32, INT32>(0, 1)	AllJoyn introspection specifies the argument with the annotation: <struct name="Point"> <field name="x" type="i"/> <field name="y" type="i"/> </struct>	{"x": 0, "y": 1}	JSON schema should indicate: "type": "object", "properties": { "x": { "type": "integer" }, "y": { "type": "integer" } }

Table 32 – Mapping from CBOR using introspection

CBOR Payload	OCF Introspection Notes	Translated AllJoyn	AllJoyn Introspection Notes
0	"type": "integer"	INT32(0)	
0	"type": "integer", "minimum": -240, "maximum": 240	INT64(0)	org.alljoyn.Bus.Type.Min = -240 org.alljoyn.Bus.Type.Max = 240
0	"type": "integer", "minimum": 0, "maximum": 248	UINT64(0)	org.alljoyn.Bus.Type.Max = 248
0.0	"type": "number"	DOUBLE(0.0)	
[1]	JSON schema indicates: "type": "array", "items": { "type": "integer", "minimum": 0, "maximum": 246 }	ARRAY<UINT64>(1)	org.alljoyn.Bus.Type.Max = 246

7 oneM2M Translation

7.1 Operational Scenarios

The purpose of the oneM2M Bridge Platform is to enable access by the oneM2M ecosystem to select OCF Servers. This is accomplished by creating Virtual OCF Clients to represent the necessary access levels to the OCF servers that are exposed to the oneM2M ecosystem. The oneM2M Bridge Platform then exposes native oneM2M entities that map to those Virtual OCF Clients.

The oneM2M Bridge Platform is an Asymmetric Client Bridge.

The mapping between the OCF data models and the oneM2M data models is specified in OCF Resource to oneM2M Module Class Mapping. Programmatic (i.e. On-the-fly) data model translation is not supported.

7.2 Enabling oneM2M Application access to OCF Servers

Each level of oneM2M application access for OCF servers is modelled as a Virtual OCF Client. In this way, oneM2M application access can be appropriately restricted and enforced by the OCF security capabilities.

7.3 Enabling OCF Client access to oneM2M Devices

This capability is not supported.

7.4 On-the-fly Translation

All devices and resources have been aligned between the OCF and oneM2M ecosystems, so on-the-fly translation is not required.

If new OCF devices are not reflected into the oneM2M ecosystem by updates to the oneM2M specifications, the Bridge Platform will not provide a successful translation of those devices.

8 Device type definitions

The required Resource Types are listed in Table 33.

Table 33 – Device type definitions

Device Name (informative)	Device Type ("rt") (Normative)	Required Resource name	Required Resource Type
Bridge	oic.d.bridge	Secure Mode	oic.r.securemode
Virtual Device	oic.d.virtual	Device	oic.wk.d

9 Resource type definitions

9.1 List of resource types

Table 34 lists the Resource Types defined in this document.

Table 34 – Alphabetical list of resource types

Friendly Name (informative)	Resource Type (rt)	Clause
AllJoyn Object	oic.r.alljoynobject	9.2
Secure Mode	oic.r.securemode	9.3

9.2 AllJoynObject

9.2.1 Introduction

This Resource is a Collection of Resources that were all derived from the same AllJoyn object.

9.2.2 Example URI

/example/AllJoynObject

9.2.3 Resource type

The Resource Type is defined as: "oic.r.alljoynobject, oic.wk.col".

9.2.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "AllJoynObject",
    "version": "2019-03-19",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": ["http"],
  "consumes": ["application/json"],
  "produces": ["application/json"],
  "paths": {
    "/example/AllJoynObject?if=oic.if.ll": {
      "get": {
        "description": "This Resource is a Collection of Resources that were all derived from the
same AllJoyn object.\n",
        "parameters": [
          {"$ref": "#/parameters/interface-all"}
        ],
        "responses": {
          "200": {
            "description": "",
            "x-example": [
              {
                "href": "/myRes1URI",
                "rt": ["x.example.widget.false"],
                "if": ["oic.if.r", "oic.if.rw", "oic.if.baseline"],
                "eps": [
                  {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
                ]
              },
              {
                "href": "/myRes2URI",
                "rt": ["x.example.widget.true"],
                "if": ["oic.if.r", "oic.if.rw", "oic.if.baseline"],
                "eps": [
                  {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
                ]
              },
              {
                "href": "/myRes3URI",
                "rt": ["x.example.widget.method1"],
                "if": ["oic.if.rw", "oic.if.baseline"],
                "eps": [
                  {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
                ]
              },
              {
                "href": "/myRes4URI",
                "rt": ["x.example.widget.method2"],
                "if": ["oic.if.rw", "oic.if.baseline"],
                "eps": [
                  {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
                ]
              }
```

```
            ],
            "schema": {
              "$ref": "#/definitions/slinks"
            }
          }
        }
      }
    },
    "/example/AllJoynObject?if=oic.if.baseline": {
      "get": {
        "description": "This Resource is a Collection of Resources that were all derived from the
same AllJoyn object.\n",
        "parameters": [
          {"$ref": "#/parameters/interface-all"}
        ],
        "responses": {
          "200": {
            "description": "",
            "x-example": {
              "rt": ["oic.r.alljoynobject", "oic.wk.col"],
              "links": [
                {
                  "href": "/myRes1URI",
                  "rt": ["x.example.widget.false"],
                  "if": ["oic.if.r", "oic.if.rw", "oic.if.baseline"],
                  "eps": [
                    {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
                  ]
                },
                {
                  "href": "/myRes2URI",
                  "rt": ["x.example.widget.true"],
                  "if": ["oic.if.r", "oic.if.rw", "oic.if.baseline"],
                  "eps": [
                    {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
                  ]
                },
                {
                  "href": "/myRes3URI",
                  "rt": ["x.example.widget.method1"],
                  "if": ["oic.if.rw", "oic.if.baseline"],
                  "eps": [
                    {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
                  ]
                },
                {
                  "href": "/myRes4URI",
                  "rt": ["x.example.widget.method2"],
                  "if": ["oic.if.rw", "oic.if.baseline"],
                  "eps": [
                    {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
                  ]
                }
              ]
            },
            "schema": {
              "$ref": "#/definitions/AllJoynObject"
            }
          }
        }
      }
    }
  },
  "parameters": {
    "interface-all": {
      "in": "query",
      "name": "if",
      "type": "string",
      "enum": ["oic.if.ll", "oic.if.baseline"]
    }
```

```
    },
    "definitions": {
      "oic.oic-link": {
        "type": "object",
        "properties": {
          "anchor": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/anchor"
          },
          "di": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/di"
          },
          "eps": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/eps"
          },
          "href": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/href"
          },
          "ins": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/ins"
          },
          "p": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/p"
          },
          "rel": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_array"
          },
          "title": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/title"
          },
          "type": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/type"
          },
          "if": {
            "description": "The OCF Interfaces supported by the target Resource",
            "items": {
              "enum": [
                "oic.if.baseline",
                "oic.if.ll",
                "oic.if.r",
                "oic.if.rw"
              ],
              "type": "string",
              "maxLength": 64
            },
            "minItems": 1,
            "uniqueItems": true,
            "type": "array"
          },
          "rt": {
            "description": "Resource Type of the target Resource",
            "items": {
              "maxLength": 64,
```



```
            "type": "string"
          },
          "minItems": 1,
          "uniqueItems": true,
          "type": "array"
        }
      },
      "required": [
        "href",
        "rt",
        "if"
      ]
    },
    "slinks" : {
      "type": "array",
      "items": {
        "$ref": "#/definitions/oic.oic-link"
      }
    },
    "AllJoynObject": {
      "type": "object",
      "properties": {
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "if": {
          "description": "The interface set supported by this resource",
          "items": {
            "enum": ["oic.if.baseline", "oic.if.ll"],
            "type": "string"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "rt": {
          "items": {
            "enum": ["oic.r.alljoynobject", "oic.wk.col"],
            "type": "string"
          },
          "maxItems": 2,
          "minItems": 2,
          "uniqueItems": true,
          "readOnly": true,
          "type": "array"
        },
        "links" : {
          "type": "array",
          "description": "A set of OCF Links.",
          "items": {
            "$ref": "#/definitions/oic.oic-link"
          }
        }
      }
    }
  }
}
```


9.2.5 Property definition

Table 35 defines the Properties that are part of the "oic.r.alljoynobject, oic.wk.col" Resource Type.

Table 35 – The Property definitions of the Resource with type "rt" = "oic.r.alljoynobject, oic.wk.col".

Property name	Value type	Mandatory	Access mode	Description
id	multiple types: see schema		Read Write	
links	array: see schema		Read Write	A set of OCF Links.
n	multiple types: see schema		Read Write	
rt	array: see schema		Read Only	
if	array: see schema		Read Only	The interface set supported by this resource
rel	multiple types: see schema	No	Read Write	
type	multiple types: see schema	No	Read Write	
if	array: see schema	Yes	Read Write	The OCF Interfaces supported by the target Resource
p	multiple types: see schema	No	Read Write	
anchor	multiple types: see schema	No	Read Write	
rt	array: see schema	Yes	Read Write	Resource Type of the target Resource
eps	multiple types: see schema	No	Read Write	
href	multiple types: see schema	Yes	Read Write	
ins	multiple types: see schema	No	Read Write	
title	multiple types: see schema	No	Read Write	
di	multiple types: see schema	No	Read Write	

9.2.6 CRUDN behaviour

Table 36 defines the CRUDN operations that are supported on the "oic.r.alljoynobject, oic.wk.col" Resource Type.

Table 36 – The CRUDN operations of the Resource with type "rt" = "oic.r.alljoynobject, oic.wk.col".

Create	Read	Update	Delete	Notify
	get			observe

9.3 SecureMode

9.3.1 Introduction

This Resource describes a secure mode on/off feature (on/off).
A secureMode value of 'true' means that the feature is on, and any Bridged Server that cannot be

communicated with securely shall not have a corresponding Virtual OCF Server, and any Bridged Client that cannot be communicated with securely shall not have a corresponding Virtual OCF Client.
A secureMode value of 'false' means that the feature is off, any Bridged Server can have a corresponding Virtual OCF Server, and any Bridged Client can have a corresponding Virtual OCF Client.

9.3.2 Example URI

/example/SecureModeResURI

9.3.3 Resource type

The Resource Type is defined as: "oic.r.securemode".

9.3.4 OpenAPI 2.0 definition


```
{
  "swagger": "2.0",
  "info": {
    "title": "SecureMode",
    "version": "2019-03-19",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LICENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": ["http"],
  "consumes": ["application/json"],
  "produces": ["application/json"],
  "paths": {
    "/example/SecureModeResURI": {
      "get": {
        "description": "This Resource describes a secure mode on/off feature (on/off).\nA secureMode value of 'true' means that the feature is on, and any Bridged Server that cannot be communicated with securely shall not have a corresponding Virtual OCF Server, and any Bridged Client that cannot be communicated with securely shall not have a corresponding Virtual OCF Client.\nA secureMode value of 'false' means that the feature is off, any Bridged Server can have a corresponding Virtual OCF Server, and any Bridged Client can have a corresponding Virtual OCF Client.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"}
        ],
        "responses": {
          "200": {
            "description": "",
            "x-example": {
              "rt": ["oic.r.securemode"],
              "secureMode": false
            },
            "schema": {
              "$ref": "#/definitions/SecureMode"
            }
          }
        }
      },
      "post": {
        "description": "Updates the value of secureMode.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": {
```

```
              "$ref": "#/definitions/SecureMode-Update"
            },
            "x-example": {
              "secureMode": true
            }
          }
        ],
        "responses": {
          "200": {
            "description": "",
            "x-example": {
              "secureMode": true
            },
            "schema": {
              "$ref": "#/definitions/SecureMode"
            }
          }
        }
      }
    }
  },
  "parameters": {
    "interface": {
      "in": "query",
      "name": "if",
      "type": "string",
      "enum": ["oic.if.rw", "oic.if.baseline"]
    }
  },
  "definitions": {
    "SecureMode": {
      "properties": {
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "if": {
          "description": "The interface set supported by this resource",
          "items": {
            "enum": ["oic.if.baseline", "oic.if.rw"],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 1,
          "readOnly": true,
          "uniqueItems": true,
          "type": "array"
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "rt": {
          "description": "Resource Type",
          "items": {
            "enum": ["oic.r.securemode"],
            "type": "string",
            "maxLength": 64
          },
          "minItems": 1,
          "uniqueItems": true,
          "readOnly": true,
          "type": "array"
        },
        "secureMode": {
          "description": "Status of the Secure Mode",
          "type": "boolean"
        }
```



```
        },
        "required":                                            ["secureMode"],
        "type":                                                      "object"
      },
      "SecureMode-Update":                                                  {
        "properties":                                                       {
          "secureMode":                                                     {
            "description":    "Status     of     the     Secure     Mode",
            "type":                                                 "boolean"
          }
        }
      }
    }
  }
}
```


9.3.5 Property definition

Table 37 defines the Properties that are part of the "oic.r.securemode" Resource Type.

Table 37 – The Property definitions of the Resource with type "rt" = "oic.r.securemode".

Property name	Value type	Mandatory	Access mode	Description
secureMode	boolean		Read Write	Status of the Secure Mode
secureMode	boolean	Yes	Read Write	Status of the Secure Mode
n	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The interface set supported by this resource
rt	array: see schema	No	Read Only	Resource Type
id	multiple types: see schema	No	Read Write	

9.3.6 CRUDN behaviour

Table 38 defines the CRUDN operations that are supported on the "oic.r.securemode" Resource Type.

Table 38 – The CRUDN operations of the Resource with type "rt" = "oic.r.securemode".

Create	Read	Update	Delete	Notify
	get	post		observe

