OCF Bridging Specification

VERSION 1.0.0 | June 2017

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 translator that exposes AllJoyn producer applications to OCF clients, and exposes OCF servers to AllJoyn consumer applications. Translation of specific AllJoyn interfaces to or from specific OCF resource types is left to other specifications. Translation of protocols other than AllJoyn is left to a future version of this specification. This document provides generic requirements that apply unless overridden by a more specific document.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. 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://allseenalliance.org/framework/documentation/learn/core/about-announcement/interface

AllJoyn Configuration Interface Specification, *Configuration Interface Definition*, Version 14.12
https://allseenalliance.org/framework/documentation/learn/core/configuration/interface

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

IEEE 754, *IEEE Standard for Floating-Point Arithmetic*, August 2008

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

IETF RFC 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 7049, *Concise Binary Object Representation (CBOR)*, October 2013
https://www.rfc-editor.org/info/rfc7049

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

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

OCF 1.0 Core Specification, *Open Connectivity Foundation Core Specification*, Version 1.0

OCF Security Specification, *Open Connectivity Foundation Security Specification*, Version 1.0

OCF ASA Mapping, *OCF Resource to ASA Interface Mapping*, v0.3 candidate, July 2016 https://workspace.openconnectivity.org/apps/org/workgroup/smarthome_tg/download.php/6287/OCF_Resource_to_ASA_Interface_Mapping_v.0.3_candidate.docx

OIC 1.1 Core Specification, *Open Interconnect Consortium Core Specification*, Version 1.1

RAML Specification, *Restful API modelling language,* Version 0.8. https://github.com/raml-org/raml-spec/blob/master/versions/raml-08/raml-08.md

OCF Resource Type Definitions, *API Definition Language for OCF Resource Type Definitions,* Release OCF-v1.0.0 https://github.com/openconnectivityfoundation/bridging

3 Terms, definitions, symbols and abbreviations

3.1 Terms and definitions

3.1.1

OCF Bridge Device

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

3.1.2

Bridged Protocol

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

3.1.3

Translator

an OCF Bridge Device component that is responsible for translating to or from a specific Bridged Protocol. More than one translator can exist on the same OCF Bridge Device, for different Bridged Protocols.

3.1.4

OCF Client

a logical entity that accesses an OCF Resource on an OCF Server, which might be a Virtual OCF Server exposed by the OCF Bridge Device.

3.1.5

Bridged Client

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

3.1.6

Virtual OCF Client

a logical representation of a Bridged Client, which an OCF Bridge Device exposes to OCF Servers.

3.1.7

Virtual Bridged Client

a logical representation of an OCF Client, which an OCF Bridge Device exposes to Bridged Servers.

3.1.8

OCF Device

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

3.1.9
Virtual OCF Server
a logical representation of a Bridged Server, which an OCF Bridge Device exposes to OCF Clients.

3.1.10
Bridged Server
a logical entity that provides data via a Bridged Protocol. For example, an AllJoyn Producer is a Bridged Server. More than one Bridged Server can exist on the same physical platform.

3.1.11
Virtual Bridged Server
a logical representation of an OCF Server, which an OCF Bridge Device exposes to Bridged Clients.

3.1.12
OCF Resource
represents an artifact modelled and exposed by the OCF Framework

3.1.13
Virtual OCF Resource
a logical representation of a Bridged Resource, which an OCF Bridge Device exposes to OCF Clients.

3.1.14
Bridged Resource
represents an artifact modelled and exposed by a Bridged Protocol. For example, an AllJoyn object is a Bridged Resource.

3.1.15
OCF Resource Property
a significant aspect or notion including metadata that is exposed through the OCF Resource

3.1.16
OCF Resource Type
an OCF Resource Property that represents the data type definition for the OCF Resource

3.1.17
Bridged Resource Type
a schema used with a Bridged Protocol. For example, AllJoyn Interfaces are Bridged Resource Types.

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

3.1.19
Onboarding Tool
defined by the OCF Security Specification as: A logical entity within a specific IoT network that establishes ownership for a specific device and helps bring the device into operational state within that network.

3.1.20
Bridged Device
a Bridged Client or Bridged Server.

3.1.21
Virtual OCF Device
a Virtual OCF Client or Virtual OCF Server.

3.1.22
CRUDN
Create Read Update Delete Notify
indicating which operations are possible on the resource

3.1.23
CSV
Comma Separated Value List
construction to have more fields in 1 string separated by commas. If a value contains a comma, then the comma can be escaped by adding "\" in front of the comma.

3.1.24
OCF
Open Connectivity Foundation
organization that created these specifications

3.1.25
RAML
RESTful API Modeling Language
Simple and succinct way of describing practically RESTful APIs (see the OIC 1.1 Core Specification, *Open Interconnect Consortium Core Specification*, Version 1.1

3.2 RAML Specification)Symbols and abbreviations

None defined.

4 Document conventions and organization

For the purposes of this document, the terms and definitions given in the OCF 1.0 Core Specification apply.

4.1 Conventions

4.2 In this specification several 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.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 this specification. 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 this specification and should be implemented. Recommended features take advantage of the capabilities of this specification, 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, 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 specification, they should not be implemented except for backward compatibility. The occurrence of a deprecated feature during operation of an implementation compliant with the current specification 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 specification.

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

Words that are emphasized are printed in *italic*.

4.3 Data types

Data types are defined in the OCF 1.0 Core Specification.

4.4 Document structure

Section 5 discusses operational scenarios. Section 6 covers generic requirements for any OCF Bridge, and section 7 covers the specific requirements for a Bridge that translates to/from AllJoyn. These are covered separately to ease the task of defining translation to other protocols in the future.

5 Operational Scenarios

5.1 Goals

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.

5.2 “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. Translators 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 translator 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 translator. 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 translator. 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 OCF ASA Mapping, and on-the-fly translation is covered in section 7.2 of this document.

5.3 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. The OIC 1.1 Core Specification makes no such requirement, but the OCF 1.0 Core Specification does. When the metadata is available, translators 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 translator, the expected interaction list is presented on the list below:

Message Type	Sender	Receiver	Metadata
Request	AllJoyn 16.10	OIC 1.1	Not available
Request	AllJoyn 16.10	OCF 1.0	Available
Request	OIC 1.1 or OCF 1.0	AllJoyn 16.10	Available
Response	AllJoyn 16.10	OIC 1.1 or OCF 1.0	Available
Response	OIC 1.1	AllJoyn 16.10	Not available
Response	OCF 1.0	AllJoyn 16.10	Available

5.4 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 above chain, payloads 2 and 4 as well as 3 and 5 should be identical.

6 OCF Bridge Device

6.1 Introduction

This section describes the functionality of an OCF Bridge Device; such a device is illustrated in Figure 1. OCF Bridge Device Components.

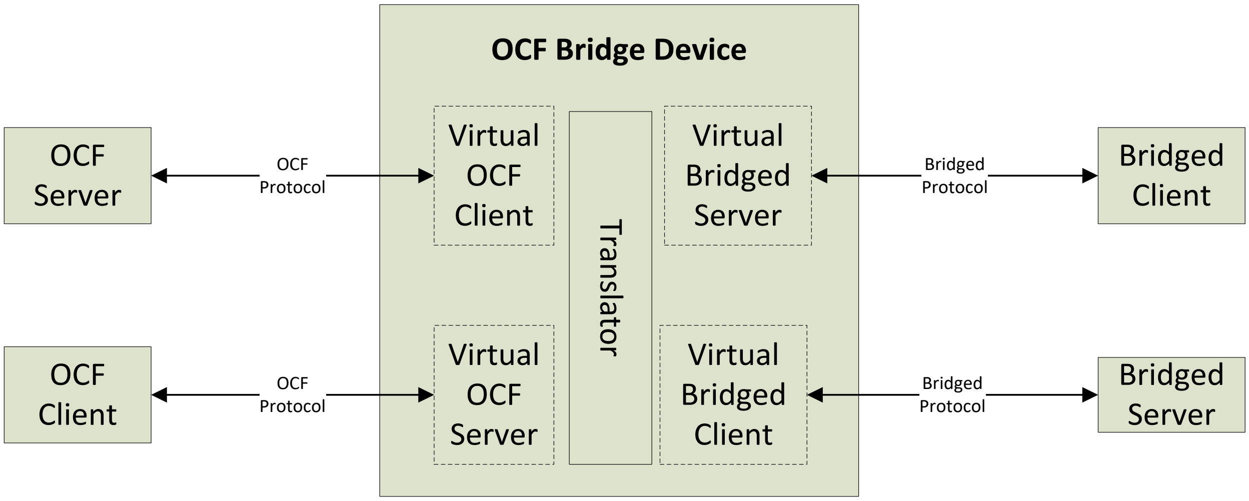

Figure 1. OCF Bridge Device ComponentsAn OCF Bridge Device is a device that represents one or more Bridged Devices as Virtual OCF Devices on the network and/or represents one or more OCF Devices as Virtual Devices using another protocol on the network. The Bridged Devices themselves are out of the scope of this document. The only difference between a native OCF Device and a Virtual Bridged Device is how the device is encapsulated in an OCF Bridge Device.

An OCF Bridge Device shall be indicated on the OCF network 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.

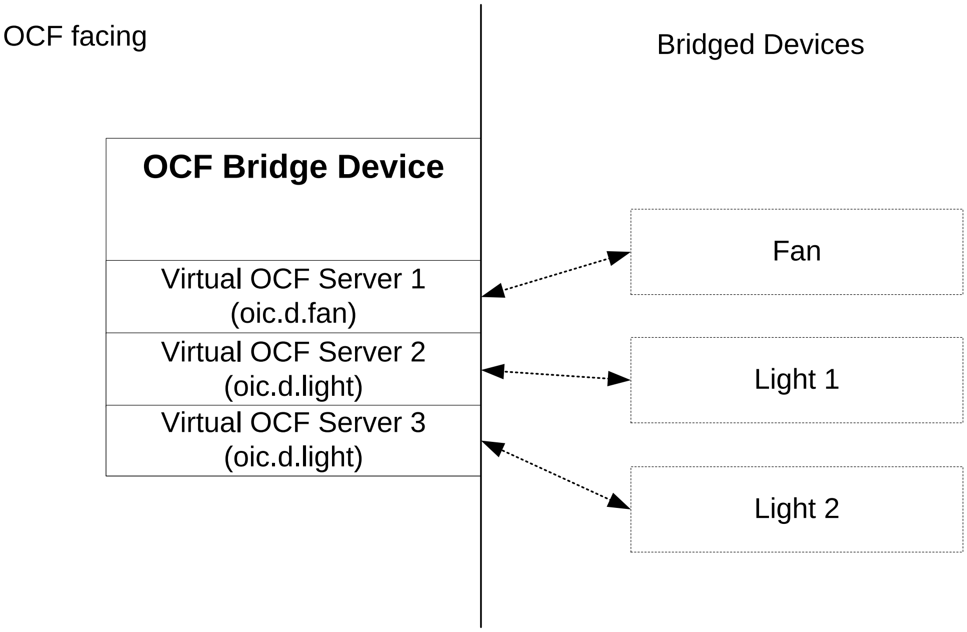
OCF facing
Bridged Devices
OCF Bridge Device
Virtual OCF Server 1 (oic.d.fan)
Virtual OCF Server 2 (oic.d.light)
Virtual OCF Server 3 (oic.d.light)
Fan
Light 1
Light 2

Figure 2: Schematic overview of an OCF Bridge Device bridging non-OCF devices

It is expected that the OCF Bridge Device creates a set of devices during the start-up of the OCF Bridge Device. The exposed set of Virtual OCF Devices can change as Bridged Devices are added or removed from the bridge. The adding and removing of Bridged Devices is implementation dependent. When an OCF Bridge Device changes the set of exposed Virtual OCF Devices, it shall notify any OCF Clients subscribed to its “/oic/res”.

6.2 Resource Discovery

An OCF Bridge Device shall detect devices that arrive and leave the Bridged network or the OCF network. Where there is no pre-existing mechanism to reliably detect the arrival and departure of devices on a network, an OCF Bridge Device 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 network. OCF Bridge Device implementations are encouraged to use a poll interval of 30 seconds plus or minus a random delay of a few seconds.

An OCF Bridge Device shall respond to network discovery commands on behalf of the exposed bridged devices. All bridged devices with all their Resources shall be listed in “/oic/res” of the Bridge. 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 shall be unique. The Bridge shall meet the requirements defined in the OCF 1.0 Core Specification for population of the Properties and Link parameters in “/oic/res”.

For example, if an OCF Bridge Device exposes Virtual OCF Servers for the fan and lights shown in Figure 2, the bridge might return the following information corresponding to the JSON below to a legacy OIC 1.1 client doing a RETRIEVE on “/oic/res”. (Note that what is returned is not in the JSON format but in a suitable encoding as defined in the OCF 1.0 Core Specification.)

```
[
  {
    "di": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "links": [
      {
        "href": "coap://[2001:db8:a::b1d4]:55555/oic/res",
        "rel": "self",
        "rt": ["oic.wk.res"],
        "if": ["oic.if.ll", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 11111}
      },
      {
        "href": "/oic/d",
        "rt": ["oic.wk.d", "oic.d.bridge"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 11111}
      },
      {
        "href": "/oic/p",
        "rt": ["oic.wk.p"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 11111}
      },
      {
        "href": "/mySecureMode",
        "rt": ["oic.r.securemode"],
        "if": ["oic.if.rw", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 11111}
      },
      {
        "href": "/oic/sec/doxm",
        "rt": ["oic.r.doxm"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 11111}
      },
      {
        "href": "/oic/sec/pstat",
        "rt": ["oic.r.pstat"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 11111}
      },
      {
        "href": "/oic/sec/cred",
        "rt": ["oic.r.cred"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 11111}
      },
      {
        "href": "/oic/sec/acl2",
        "rt": ["oic.r.acl2"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 11111}
      },
      {
        "href": "/myIntrospection",
        "rt": ["oic.wk.introspection"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 11111}
      }
    ]
  },
  {
    "di": "88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
    "links": [
      {
        "href": "coaps://[2001:db8:a::b1d4]:22222/oic/res",
        "rt": ["oic.wk.res"],
        "if": ["oic.if.ll", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 22222}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:22222/oic/d",
        "rt": ["oic.wk.d", "oic.d.fan", "oic.d.virtual"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 22222}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:22222/oic/p",
        "rt": ["oic.wk.p"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 22222}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:22222/myFan",
        "rt": ["oic.r.switch.binary"],
        "if": ["oic.if.a", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 22222}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:22222/oic/sec/doxm",
        "rt": ["oic.r.doxm"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 22222}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:22222/oic/sec/pstat",
        "rt": ["oic.r.pstat"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 22222}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:22222/oic/sec/cred",
        "rt": ["oic.r.cred"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 22222}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:22222/oic/sec/acl2",
        "rt": ["oic.r.acl2"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 22222}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:22222/myFanIntrospection",
        "rt": ["oic.wk.introspection"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 22222}
      }
    ]
  },
  {
    "di": "dc70373c-1e8d-4fb3-962e-017eaa863989",
    "links": [
      {
        "href": "coaps://[2001:db8:a::b1d4]:33333/oic/res",
        "rt": ["oic.wk.res"],
        "if": ["oic.if.ll", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 33333}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:33333/oic/d",
        "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 33333}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:33333/oic/p",
        "rt": ["oic.wk.p"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 33333}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:33333/myLight",
        "rt": ["oic.r.switch.binary"],
        "if": ["oic.if.a", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 33333}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:33333/oic/sec/doxm",
        "rt": ["oic.r.doxm"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 33333}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:33333/oic/sec/pstat",
        "rt": ["oic.r.pstat"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 33333}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:33333/oic/sec/cred",
        "rt": ["oic.r.cred"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 33333}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:33333/oic/sec/acl2",
        "rt": ["oic.r.acl2"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 33333}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:33333/myLightIntrospection",
        "rt": ["oic.wk.introspection"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 33333}
      }
    ]
  },
  {
    "di": "8202138e-aa22-452c-b512-9ebad02bef7c",
    "links": [
      {
        "href": "coaps://[2001:db8:a::b1d4]:44444/oic/res",
        "rt": ["oic.wk.res"],
        "if": ["oic.if.ll", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 44444}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:44444/oic/d",
        "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 44444}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:44444/oic/p",
        "rt": ["oic.wk.p"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 44444}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:44444/myLight",
        "rt": ["oic.r.switch.binary"],
        "if": ["oic.if.a", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 44444}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:44444/oic/sec/doxm",
        "rt": ["oic.r.doxm"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 44444}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:44444/oic/sec/pstat",
        "rt": ["oic.r.pstat"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 44444}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:44444/oic/sec/cred",
        "rt": ["oic.r.cred"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 44444}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:44444/oic/sec/acl2",
        "rt": ["oic.r.acl2"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 44444}
      },
      {
        "href": "coaps://[2001:db8:a::b1d4]:44444/myLightIntrospection",
        "rt": ["oic.wk.introspection"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 44444}
      }
    ]
  }
]
```

The above example illustrates that each Virtual OCF Server has its own “di” and endpoint exposed by the bridge, and that “/oic/p” and “/oic/d” are available for each Virtual OCF Server.

When an OCF Client requests a content format of “application/vnd.ocf+cbor”, the same bridge will return information corresponding to the JSON below. (Note that what is returned is not in the JSON format but in a suitable encoding as defined in the OCF 1.0 Core Specification.)

```
[
 {
   "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": "/mySecureMode",
   "rt": ["oic.r.securemode"],
   "if": ["oic.if.rw", "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"}]
 },


 {
   "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"}]
   },

   {
     "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"}]
   },

   {
     "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"}]
   }
  ]
```

6.3 General Requirements

The translator 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 translator shall stop translating any Bridged Protocol device exposed in OCF via another translator if the translator sees the device via the Bridged Protocol. Similarly, the translator shall not advertise an OCF Device back into OCF, and the translator shall stop translating any OCF device exposed in the Bridged Protocol via another translator if the translator sees the device via OCF. These require that the translator can determine when a device is already being translated. A Virtual OCF Device shall be indicated on the OCF network with a Device Type of “oic.d.virtual”. This allows translators to determine if a device is already being translated when multiple translators are present. How a translator determines if a device is already being translated on a non-OCF network is described in the protocol-specific sections below.

Each Bridged Server shall be exposed as a separate Virtual OCF Server, with its own endpoint, and its own “/oic/d” and “/oic/p”. The Virtual OCF Server’s “/oic/res” resource would be the same as for any ordinary OCF Server that uses a resource directory. That is, it does not respond to multicast discovery requests (because the OCF Bridge Device responds on its behalf), but a unicast query elicits a response listing its own resources with a “rel”=“hosts” relationship, and an appropriate “anchor” to indicate that it is not the OCF Bridge Device itself. This allows platform-specific, device-specific, and resource-specific fields to all be preserved across translation.

6.4 Security

6.4.1 General Security Considerations

The OCF Bridge Device shall go through OCF ownership transfer as any other onboardee would. Separately, it shall go through the Bridged Protocol’s ownership transfer mechanism (e.g., AllJoyn claiming) normally as any other onboardee would.

The OCF Bridge Device shall be field updatable. (This requirement need not be tested but can be certified via a vendor declaration.)

Unless an administrator opts in to allow it (see section 9.2), a translator shall not expose connectivity to devices that it cannot get a secure connection to.

Each Virtual OCF Device shall be provisioned for security by an OCF Onboarding tool. Each Virtual Bridged Device should be provisioned as appropriate in the Bridged ecosystem. In other words, Virtual Devices are treated the same way as physical Devices. They are entities that have to be provisioned in their network.

The Translator shall provide a “piid” value that can be used to correlate a non-OCF Device with its corresponding Virtual OCF Device, as specified in Section 6.3. An Onboarding Tool might use this correlation to improve the Onboarding user experience by eliminating or reducing the need for user input, by automatically creating security settings for Virtual OCF Devices that are equivalent to the security settings of their corresponding non-OCF Devices. See the OCF Security Specification for detailed information about Onboarding.

Each Virtual Device shall implement the security requirements of the ecosystem that it is connected to. For example, each Virtual OCF Device shall implement the behaviour required by the OCF 1.0

Core Specification and the OCF Security Specification. Each Virtual OCF Device shall perform authentication, access control, and encryption according to the security settings it received from the Onboarding Tool.

Depending on the architecture of the Translator, authentication and access control might take place just within each ecosystem, but not within the Translator. For example, when an OCF Client sends a request to a Virtual OCF Server:

- Authentication and access control might be performed by the Virtual OCF Server when receiving the request from the OCF Client.
- The Translator might not perform authentication or access control when the request travels through the Translator to the corresponding Virtual Bridged Client.
- Authentication and access control might be performed by the target Bridged Server when it receives the request from the Virtual Bridged Client, according to the security model of the Bridged ecosystem.

A Translator may receive unencrypted data coming from a Bridged Client through a Virtual Bridged Device. The translated message shall be encrypted by the corresponding Virtual OCF Client, before sending it to the target OCF Device, if this OCF Device requires encryption.

A Translator may receive unencrypted data coming from an OCF Client through a Virtual OCF Server. After translation, this data shall be encrypted by the corresponding Virtual Bridged Client, before sending it to the target Bridged Server, if this Bridged Server requires encryption.

A Translator shall protect the data while that data travels between a Virtual Client and a Virtual Server, through the Translator. For example, if the Translator sends data over a network, the Translator shall perform appropriate authentication and access control, and shall encrypt the data, between all peers involved in this communication.

6.4.2 Blocking communication of Bridged Devices with the OCF ecosystem

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.

In addition, an OCF Onboarding Tool can block the communication of a particular Virtual OCF Client with all OCF Servers, or block the communication of all OCF Clients with a particular Virtual OCF Server, in the same way as it would for any other OCF Device. See section 8.5 of the OCF Security Specification for information about the soft reset state.

7 AllJoyn Translation

7.1 Requirements Specific to an AllJoyn Translator

7.1.1 Introduction

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

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

7.1.2 Exposing AllJoyn producer devices to OCF Clients

7.1.2.1 Virtual OCF Devices and Resources

As specified in the OCF Security Specification, the value of the “di” property of OCF Devices (including Virtual OCF Devices) shall be established as part of Onboarding of that Virtual OCF Device.

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 (as discussed below), 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. 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 Section 9.3, and the items in the collection are the Resources with the translated Resource Types as discussed below.

The value of the “piid” property of “/oic/d” for each Virtual OCF Device 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 Translator 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 section 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 section 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 translator 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 translator 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.

- If the AllJoyn interface is in a well-defined set (defined in OCF ASA Mapping or section 7.1.2.2 below) of interfaces where standard forms exist on both the AllJoyn and OCF sides, the translator shall either:
    a. follow the specification for translating that interface specially, or
    b. not translate the AllJoyn interface.
- If the AllJoyn interface is not in the well-defined set, the translator shall either:
    a. not translate the AllJoyn interface, or
    b. algorithmically map the AllJoyn interface as specified in section 7.2 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 below.
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 the table below. 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.)

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 “const”, even if not specified in introspection XML.
- Resource Types mapping AllJoyn Properties with access “readwrite” shall support the “oic.if.rw” Interface. Resource Types mapping AllJoyn Properties with access “read” shall support the “oic.if.r” Interface. Resource Types supporting both the “oic.if.rw” and “oic.if.r” 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” Interface. Each argument of the AllJoyn Method is mapped to a separate Property on the Resource Type, where the name of that Property is prefixed with the AllJoyn Method name in order to help get uniqueness across all Resource Types on the same Resource. When the AllJoyn argument name is not specified, the name of the property shall be “x.<AllJoynInterfaceName>.<MethodName>arg<#>”, where <AllJoynInterfaceName> is transformed as described above and <#> is the 0-indexed position of the argument in the AllJoyn introspection XML, prefixed with the AllJoyn Method name. In addition, that Resource Type has an extra “x.<AllJoynInterfaceName>.<MethodName>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” 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 the AllJoyn Signal name in order to help get uniqueness across all Resource Types on the same Resource. When the AllJoyn argument name is not specified, the name of the property shall be “x.<AllJoynInterfaceName>.<SignalName>arg<#>”, where <AllJoynInterfaceName> is transformed as described above and <#> is the 0-indexed position of the argument in the AllJoyn introspection XML, prefixed with the AllJoyn Signal name. In addition, that Resource Type has an extra “x.<AllJoynInterfaceName>.<SignalName>validity” property (also prefixed with the Signal name) 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 Section 7.2.

If an AllJoyn operation fails, the translator shall send an appropriate OCF error response to the OCF client. If an AllJoyn error name is available, 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>". The <error name> shall be taken from the AllJoyn error name field, with the "org.openconnectivity.Error." prefix removed if it is present. If the resulting error name is of the form "<#>" where <#> is an error code without a decimal (e.g., "404"), the error code shall be the error code indicated by the error name. Example: "org.openconnectivity.Error.404" becomes "404", which shall result in an error 4.04 for a CoAP transport.

7.1.2.2 Exposing an AllJoyn producer application as a Virtual OCF Server

Table 1 shows how OCF Device properties, as specified in Table 20 in the OCF 1.0 Core Specification, shall be derived, typically from fields specified in the AllJoyn About Interface Specification and AllJoyn Configuration Interface Specification.

Table 1: oic.wk.d resource type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mand?	From AJ Field name	AJ Description	AJ Mand?
(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 the core specification this device is implemented to, The syntax is "core.major.minor”]	Y	(none)	Translator should return its own value	
Device ID	di	Unique identifier for Device. This value shall be as defined in Table 5 for DeviceID.	Y	(none)	Use as defined in the OCF Security Specification	

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 including any null terminator) and the AppId and using the algorithm in IETF RFC 4122 section 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.	Peer GUID: The peer GUID is the only persistent identity for a peer. Peer GUIDs are used by the authentication mechanisms to uniquely and identify a remote application instance. The peer 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.	Peer GUID: conditionally Y DeviceId: Y AppId: Y
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. Each value is formatted as "<interface name>.<Version property value>".	This specification 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 2 shows how OCF Device Configuration properties, as specified in Table 15 in the OCF 1.0 Core Specification, shall be derived:

Table 2: oic.wk.con resource type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mand?	From AJ Field name	AJ Description	AJ Mand?
(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 text shall not start with a quote (").	N	org.openconnectivity.r (if it exists, else property shall be absent)		N
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	N	AppName	Application name assigned by the app manufacturer (developer or the OEM).	Y

		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.				
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 3 shows how OCF Platform properties, as specified in Table 21 in the OCF 1.0 Core Specification, are derived, typically from fields specified in the AllJoyn About Interface Specification and AllJoyn Configuration Interface Specification.

Table 3: oic.wk.p Resource Type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mand?	From AJ Field name	AJ Description	AJ Mand?
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 section 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 the app is running.	N
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 4 shows how OCF Platform Configuration properties, as specified in Table 16 in the OCF 1.0 Core Specification, shall be derived:

Table 4: oic.wk.con.p Resource Type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mand?	From AJ Field name	AJ Description	AJ Mand?
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.

7.1.3 Exposing OCF resources to AllJoyn consumer applications

7.1.3.1 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 translators to determine if a device is already being translated when multiple translators 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 on the resource shall be the OCF URI path. The AllJoyn About data shall be populated per Table 5 below.

A translator implementation is encouraged to maintain a cache of OCF resources to handle WhoImplements queries from the AllJoyn side, and emit an Announce Signal for each OCF Server. Specifically, the translator 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.

- If the Resource Type is in a well-defined set (defined in OCF ASA Mapping or section 7.1.3.2 below) of resource types where standard forms exist on both the AllJoyn and OCF sides, the translator shall either:
    a. follow the specification for translating that resource type specially, or
    b. not translate the Resource Type.
- If the Resource Type is not in the well-defined set (but is not a Device Type), the translator shall either:
    a. not translate the Resource Type, or
    b. algorithmically map the Resource Type as specified in section 7.2 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 name 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”).
c. Else, convert the hyphen to an underscore (i.e., convert “-” to “_”).

Some examples are shown in the table below. 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.)

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.
- 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 in OIC 1.1 but can be determined in OCF 1.0 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” 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 Section 7.2.

If an OCF operation fails, the translator 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 error name and error message shall be extracted from the error message. Otherwise, the error name shall be "org.openconnectivity.Error.<#>" where <#> is the error code without a decimal (e.g., "404").

7.1.3.2 Exposing an OCF server as a Virtual AllJoyn Producer

Table 5 shows how AllJoyn About Interface fields are derived, based on properties in “oic.wk.d”, “oic.wk.con”, “oic.wk.p”, or “oic.wk.con.p”.

Table 5: AllJoyn About Data fields

To AJ Field name	AJ Description	AJ Mand?	From OCF Property title	OCF Property name	OCF Description	OCF Mand?
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 translator 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, [{“language”:”en”, “value”:”Dave’s Laptop”}]	N
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	ln 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	N (ln), Y (n)

To AJ Field name	AJ Description	AJ Mand?	From OCF Property title	OCF Property name	OCF Description	OCF Mand?
					Name (n) value shall be included in this array.	
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	ln	If ln 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)		Translator 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

To AJ Field name	AJ Description	AJ Mand?	From OCF Property title	OCF Property name	OCF Description	OCF Mand?
org.openconnectivity.mnos		N	OS Version	mnos (if it exists, else field shall be absent)	Version of platform resident OS – string (defined by manufacturer)	N
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

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 6 shows how AllJoyn Configuration Interface fields shall be derived, based on properties in “oic.wk.con” or “oic.wk.con.p”.

Table 6: AllJoyn Configuration Data fields

To AJ Field name	AJ Description	AJ Mand?	From OCF Property title	OCF Property name	OCF Description	OCF Mand?
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 language unless the property specifies otherwise.	N
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

To AJ Field name	AJ Description	AJ Mand?	From OCF Property title	OCF Property name	OCF Description	OCF Mand?
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

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.

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

7.2.1 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.

7.2.2 Translation without aid of introspection

7.2.2.1 Introduction

This section describes how translators shall translate messages between the two payload formats in the absence of introspection metadata from the actual device. This situation arises in the following cases:

- Requests to OIC 1.1 devices
- Replies from OIC 1.1 devices
- Content not described by introspection, such as the inner payload of AllJoyn properties of type “D-Bus VARIANT”.

Since introspection is not available, the translator 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 below in terms of those generic types.

7.2.2.2 Booleans

Boolean conversion is trivial since both sides support this type.

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

7.2.2.3 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.

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)(1)	
“q” - INT16 (signed 16-bit)	
“”" - INT32 (signed 32-bit)	
”x” - INT64 (signed 64-bit)(1)	
“d” - DOUBLE (IEEE 754 double precision)	

From JSON type	To D-Bus type
Number	“d” - DOUBLE(2)

Notes and rationales:

1. 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 the OCF 1.0 Core Specification does not allow for integral numbers outside the range $-2^{53} \leq x \leq 2^{53}$.
2. To provide the most predictable result, all translations from OCF to AllJoyn produce values of type “d” DOUBLE (IEEE 754 double precision).

7.2.2.4 Text strings

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 translators not use CBOR indeterminate text strings (first byte 0x7f).

7.2.2.5 Byte arrays

The translation of a byte array is direction-specific.

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

The base64url encoding is specified in IETF RFC 4648 section 5.

7.2.2.6 D-Bus Variants

D-Bus type	JSON type
“v” - VARIANT	*see below*

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.

7.2.2.7 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). In the reverse direction, Section 7.2.2.4 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.

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.

7.2.2.8 D-Bus Structures

The translation of the following types is direction-specific:

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.

7.2.2.9 Arrays

The translation of the following types is bidirectional:

D-Bus type	JSON type
“ay” - ARRAY of BYTE	(base64-encoded) string – see Section 7.2.2.5
“ae” - ARRAY of DICT_ENTRY	object – see Section 7.2.2.10

The translation of the following types is direction-specific:

From D-Bus type	To JSON type
“a” – ARRAY of anything else not specified above	Array

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

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.

Conversion of D-Bus arrays of variants uses the conversion of variants as specified above, 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.

7.2.2.10 Dictionaries / Objects

D-Bus type	JSON type
“a{sv}” - dictionary of STRING to VARIANT	Object

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.

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

7.2.2.11 Non-translatable types

D-Bus Type
“h” – UNIX_FD (Unix file descriptor)

JSON type
null
undefined (not officially valid JSON, but some implementations permit it)

The above types are not translatable, and the translator should drop the incoming message. None of the types above are in current use by either AllJoyn, OIC 1.1, or future OCF 1.0 devices, so the inability to translate them should not be a problem.

7.2.2.12 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

Source D-Bus	JSON Result
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”

1377

1378

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”)

Source JSON	D-Bus Result
[]	ARRAY<VARIANT>()
[1]	ARRAY<IDOUBLE>(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” → DOUBLE(1.0))
{“1”: 1}	map<STRING, VARIANT>(“1” → DOUBLE(1.0))

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

Note:

1. 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.

7.2.3 Translation with aid of introspection

7.2.3.1 Introduction to Introspection Metadata

When introspection is available, the translator 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 translator 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 Section 12.3 of the OCF 1.0 Core Specification, JSON format attribute values are as defined in JSON Schema Validation, and JSON media attribute values are as defined in JSON Hyper-Schema.

7.2.3.2 Translation of the introspection itself

Note that both OCF 1.0 and AllJoyn require all services exposed to include introspection metadata, which means the translator will need to translate the introspection information on-the-fly for each OCF resource or AllJoyn producer it finds. The translator 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.

7.2.3.3 Variability of introspection data

Introspection data is not a constant and the translator 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 translator 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 translator by “on-the-fly” translation will only be accessed by generic client applications. Dedicated applications will only use “deep binding” translation.

7.2.3.4 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 translator 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 translator 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 translator shall use the following rules.

The following table 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.

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)

The following table indicates how to translate from a D-Bus type to the corresponding JSON type.

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 format attribute “uint64”	IETF RFC 7159 section 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 format attribute “int64”	IETF RFC 7159 section 6 explains that other JSON integers are not interoperable.
“d” (double)	number	

7.2.3.5 Text string and byte arrays

D-Bus Type	**JSON type**	**JSON media attribute, binaryEncoding property**
“s” – STRING	String	(none)
“ay” - ARRAY of BYTE	String	base64

There’s no difference in the translation of text strings and byte arrays compared to the previous section. This section simply lists the JSON equivalent types for the generated OCF introspection.

In addition, the mapping of the following JSON Types is direction-specific:

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 this table above shall be treated the same as if the format and pattern attributes were absent, by simply mapping the value to a D-Bus string.

7.2.3.6 D-Bus Variants

D-Bus Type	JSON Type
“v” – VARIANT	*see below*

If the introspection of an AllJoyn producer indicates a value in a request should be a D-Bus VARIANT, the translator 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.

7.2.3.7 D-Bus Object Paths and Signatures

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

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 translator 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.

7.2.3.8 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 translator 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 translator shall use a JSON object to represent a structure, mapping each member to the entry with that name. The translator 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 translator shall follow the rule for translating D-Bus structures without the aid of introspection data.

7.2.3.9 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 translator shall apply the constraining or relaxing rules specified in other sections.

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”).

7.2.3.10 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.

7.2.3.11 Examples

AllJoyn Source	AllJoyn Introspection Notes	Translated JSON Payload	OCF Introspection Notes

UINT32 (0)		0	JSON schema should indicate:
			type = integer,
			minimum = 0
			maximum = 4294967295

INT64 (0)		0	Since no Min/Max annotations exist in AllJoyn, JSON schema should indicate:
			type = string
			pattern = ^0|(-?[1-9][0-9]{0,18})}$

UINT64 (0)		"0"	Since no Max annotation exists in AllJoyn, JSON schema should indicate:
			type = string
			pattern = ^0|([1-9][0-9]{0,19})$

STRING("Hello")		"Hello"	JSON schema should indicate: type = string
OBJECT_PATH("/")		"/"	JSON schema should indicate: type = string
SIGNATURE("g")		"g"	JSON schema should indicate: type = string
ARRAY<BYTE>(0x48, 0x65, 0x6c, 0x6c, 0x6f)		"SGVsbG8"	JSON schema should indicate: type = string media binaryEncoding = base64
VARIANT(*anything)*		*?*	JSON schema should indicate: type = value

ARRAY<INT32>()		[]	JSON schema should indicate:
			type = array
			items = integer

ARRAY<INT64>()		[]	JSON schema should indicate:
			type = array
			items = string
			items.pattern = ^0|([1-9][0-9]{0,18})$

STRUCT< INT32, INT32>(
0, 1
	AllJoyn introspection specifies the argument with the annotation:	["x": 0, "y": 1]	JSON schema should indicate:
			type = object
			element.x.type = integer

AllJoyn Source	AllJoyn Introspection Notes	Translated JSON Payload	OCF Introspection Notes

)	`<struct name="Point">`
	`  <field name="x" type="i"/>`
	`  <field name="y" type="i"/>`
	`</struct>`
			element.y.type = integer

1475

CBOR Payload	OCF Introspection Notes	Translated AllJoyn	AllJoyn Introspection Notes
0	JSON type is “integer”	INT32(0)	

0	JSON type is “integer”
	minimum = -2^{40}
	maximum = 2^{40}
		INT64(0)	org.alljoyn.Bus.Type.Min = -2^{40}
			org.alljoyn.Bus.Type.Max = 2^{40}

0	JSON type is “integer”
	minimum = 0
	maximum = 2^{48}
		UINT64(0)	org.alljoyn.Bus.Type.Max = 2^{48}

0.0	JSON type is “float”	DOUBLE(0.0)	

[1]	JSON schema indicates:
	type = array
	items = integer
	element.minimum = 0
	element.maximum = 2^{46}
		ARRAY<UINT64>(1)	org.alljoyn.Bus.Type.Max = 2^{46}

1476
8 Device Type Definitions

1477 The required Resource Types are listed in the table below.

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

All the sections in 9 describe the Resource Types with a restful API definition language. The Resource Type definitions presented in 9 are formatted for readability, and so may appear to have extra line breaks. The contents of the Resource Types without the extra line breaks are available in OCF Resource Type Definitions.

Table 7 Alphabetical list of resource types

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

9.2 Secure Mode

9.2.1 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, and:

- any Bridged Server can have a corresponding Virtual OCF Server, and
- any Bridged Client can have a corresponding Virtual OCF Client.

9.2.2 Example URI Path

/example/SecureModeResURI

9.2.3 Resource Type

The resource type (rt) is defined as: oic.r.securemode.

9.2.4 RAML Definition

```
#%RAML 0.8
title: OCFSecureMode
version: v1.0.0-20170531
traits:
 - interface:
     queryParameters:
       if:
         enum: ["oic.if.rw", "oic.if.baseline"]


/example/SecureModeResURI:
  description: |
    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.

  is: ['interface']

  get:
    description: |
      Retrieves the value of secureMode.
    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "id": "https://www.openconnectivity.org/ocf-
apis/bridging/schemas/oic.r.securemode.json#",
                "$schema": "http://json-schema.org/draft-04/schema#",
                "description" : "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
reserved.",
                "title": "Secure Mode",
                "definitions": {
                  "oic.r.securemode": {
                    "type": "object",
                    "properties": {
                      "secureMode": {
                        "type": "boolean",
                        "description": "Status of the Secure Mode"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  {"$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core"},
                  {"$ref": "#/definitions/oic.r.securemode"}
                ],
                "required": [ "secureMode" ]
              }
            example: |
              {
                "rt":          ["oic.r.securemode"],
                "id":          "unique_example_id",
                "secureMode":  false
              }

  post:
    description: |
      Updates the value of secureMode.
    body:
      application/json:
        schema: |
          {
            "id": "https://www.openconnectivity.org/ocf-
apis/bridging/schemas/oic.r.securemode.json#",
            "$schema": "http://json-schema.org/draft-04/schema#",
            "description" : "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
reserved.",
            "title": "Secure Mode",
            "definitions": {
              "oic.r.securemode": {
                "type": "object",
                "properties": {
                  "secureMode": {
                    "type": "boolean",
                    "description": "Status of the Secure Mode"
                  }
                }
              }
            },
            "type": "object",
            "allOf": [
              {"$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core"},
              {"$ref": "#/definitions/oic.r.securemode"}
            ],
            "required": [ "secureMode" ]
          }
        example: |
          {
            "id":          "unique_example_id",
            "secureMode":  true
          }
    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "id": "https://www.openconnectivity.org/ocf-
apis/bridging/schemas/oic.r.securemode.json#",
                "$schema": "http://json-schema.org/draft-04/schema#",
                "description" : "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
reserved.",
                "title": "Secure Mode",
                "definitions": {
                  "oic.r.securemode": {
                    "type": "object",
                    "properties": {
                      "secureMode": {
                        "type": "boolean",
                        "description": "Status of the Secure Mode"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  {"$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core"},
                  {"$ref": "#/definitions/oic.r.securemode"}
                ],
                "required": [ "secureMode" ]
              }
            example: |
              {
                "id":          "unique_example_id",
                "secureMode":  true
              }
```

9.2.5 Swagger2.0 Definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "OCFSecureMode",
    "version": "v1.0.0-20170531",
    "license": {
      "name": "copyright 2016-2017 Open Connectivity Foundation, Inc. All rights reserved.",
      "x-description": "Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:\n        1.
Redistributions of source code must retain the above copyright notice, this list of conditions and
the following disclaimer.\n        2.  Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer in the documentation and/or
other materials provided with the distribution.\n\n        THIS SOFTWARE IS PROVIDED BY THE Open
Connectivity Foundation, INC. \"AS IS\" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR
WARRANTIES OF NON-INFRINGEMENT, ARE DISCLAIMED.\n        IN NO EVENT SHALL THE Open Connectivity
Foundation, INC. OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)\n        HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF SUCH DAMAGE.\n"
    }
  },
  "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.\nRetrieves the value of secureMode.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"}
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example":
                {
                  "rt":          ["oic.r.securemode"],
                  "id":          "unique_example_id",
                  "secureMode":  false
                }
                ,
              "schema": { "$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" },
            "x-example":
              {
                "id":          "unique_example_id",
                "secureMode":  true
              }
          }
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example":
                {
```

```
                "id":          "unique_example_id",
                "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": {
          "description": "Instance ID of this specific resource",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "if": {
          "description": "The interface set supported by this resource",
          "items": {
            "enum": [
              "oic.if.baseline",
              "oic.if.ll",
              "oic.if.b",
              "oic.if.lb",
              "oic.if.rw",
              "oic.if.r",
              "oic.if.a",
              "oic.if.s"
            ],
            "type": "string"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "n": {
          "description": "Friendly name of the resource",
          "maxLength": 64,
          "readOnly": true,
          "type": "string"
        },
        "rt": {
          "description": "Resource Type",
          "items": {
            "maxLength": 64,
            "type": "string"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "secureMode": {
          "description": "Status of the Secure Mode",
          "type": "boolean"
        }
      },
      "required": [
        "secureMode"
      ],
        "type": "object"
      }

    }
  }
```

9.2.6 Property Definition

Property name	Value type	Mandatory	Access mode	Description
secureMode	boolean	Yes	Read Write	Status of the Secure Mode

9.2.7 CRUDN behaviour

Example Resource URI	Create	Read	Update	Delete	Notify
/example/SecureModeResURI		get	post		get

9.3 AllJoyn Object

9.3.1 Introduction

This resource is a collection of resources that were all derived from the same AllJoyn object.

9.3.2 Example URI Path

/example/AllJoynObject/

9.3.3 Resource Type

The resource type (rt) is defined as: oic.r.alljoynobject.

9.3.4 RAML Definition

```
#%RAML 0.8
title: OCFAllJoynObject
version: v1.0.0-20170531
traits:
 - interface-baseline:
     queryParameters:
       if:
         enum: ["oic.if.baseline"]
 - interface-ll:
     queryParameters:
       if:
         enum: ["oic.if.ll"]


/example/AllJoynObject/?if=oic.if.baseline:
  description: |
    This resource is a collection of resources that were all derived from the same AllJoyn object.

  is: ['interface-baseline']

  get:
    description: |
      Retrieves the current AllJoyn object information.
    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "id": "https://www.openconnectivity.org/ocf-
apis/bridging/schemas/oic.r.alljoynobject.json#",
                "$schema": "http://json-schema.org/draft-04/schema#",
                "description" : "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
reserved.",
                "title": "AllJoyn Object",
                "definitions": {
                  "oic.r.alljoynobject": {
                    "type": "object",
                    "allOf": [
                      {
                        "$ref": "../../core/schemas/oic.collection-
schema.json#/definitions/oic.collection"
                      },
                      {
                        "properties": {
                          "rt": {
                            "type": "array",
                            "minItems": 2,
                            "maxItems": 2,
                            "uniqueItems": true,
                            "items": {
                              "enum": ["oic.r.alljoynobject","oic.wk.col"]
                            }
                          }
                        }
                      }
                    ]
                  }
                },
                "type": "object",
                "allOf": [
                  {"$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core"},
                  {"$ref": "#/definitions/oic.r.alljoynobject"}
                ]
              }
            example: |
              {
                "rt":    ["oic.r.alljoynobject","oic.wk.col"],
                "id":    "unique_example_id",
                "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"}]}
                ]
              }
/example/AllJoynObject/?if=oic.if.ll:
  description: |
    This resource is a collection of resources that were all derived from the same AllJoyn object.

  is: ['interface-ll']

  get:
    description: |
      Retrieves the Links in the current AllJoyn object.
    responses:
      200:
        body:
          application/json:
            schema: |
              {
                "id": "https://www.openconnectivity.org/ocf-
apis/bridging/schemas/oic.r.alljoynobject-ll#",
                "$schema": "http://json-schema.org/draft-04/schema#",
                "description" : "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
reserved.",
                "title": "AllJoyn Object Links List Schema",
                "definitions": {
                  "oic.r.alljoynobject-ll": {
                    "allOf": [
                      {
                        "$ref": "../../core/schemas/oic.collection.linkslist-
schema.json#/definitions/oic.collection.alllinks"
                      }
                    ]
                  }
                },
                "allOf": [
                  {"$ref": "#/definitions/oic.r.alljoynobject-ll"}
                ]
              }
            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"}]}
              ]


```

9.3.5 Swagger2.0 Definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "OCFAllJoynObject",
    "version": "v1.0.0-20170531",
    "license": {
      "name": "copyright 2016-2017 Open Connectivity Foundation, Inc. All rights reserved.",
      "x-description": "Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:\n        1.
Redistributions of source code must retain the above copyright notice, this list of conditions and
the following disclaimer.\n        2.  Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer in the documentation and/or
other materials provided with the distribution.\n\n        THIS SOFTWARE IS PROVIDED BY THE Open
Connectivity Foundation, INC. \"AS IS\" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR
WARRANTIES OF NON-INFRINGEMENT, ARE DISCLAIMED.\n        IN NO EVENT SHALL THE Open Connectivity
Foundation, INC. OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)\n        HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF SUCH DAMAGE.\n"
    }
  },
  "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.\nRetrieves the Links in the current AllJoyn object.\n",
        "parameters": [
          {"$ref": "#/parameters/interface-ll"}
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example":
                [
                  {"href": "/myRes1URI", "rt": ["x.example.widget.false"], "if":
["oic.if.r","oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
                  {"href": "/myRes2URI", "rt": ["x.example.widget.true"], "if":
["oic.if.r","oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
                  {"href": "/myRes3URI", "rt": ["x.example.widget.method1"], "if":
["oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
                  {"href": "/myRes4URI", "rt": ["x.example.widget.method2"], "if":
["oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]}
                ]
                ,
              "schema": { "$ref": "#/definitions/AllJoynObject-ll" }
            }
        }
      }
    },
    "/example/AllJoynObject/?if=oic.if.baseline" : {
      "get": {
        "description": "This resource is a collection of resources that were all derived from the
same AllJoyn object.\nRetrieves the current AllJoyn object information.\n",
        "parameters": [
          {"$ref": "#/parameters/interface-baseline"}
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example":
                {
                  "rt":    ["oic.r.alljoynobject","oic.wk.col"],
                  "id":    "unique_example_id",
                  "links": [
                    {"href": "/myRes1URI", "rt": ["x.example.widget.false"], "if":
["oic.if.r","oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
                    {"href": "/myRes2URI", "rt": ["x.example.widget.true"], "if":
["oic.if.r","oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
                    {"href": "/myRes3URI", "rt": ["x.example.widget.method1"], "if":
["oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
                    {"href": "/myRes4URI", "rt": ["x.example.widget.method2"], "if":
["oic.if.rw","oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]}
                  ]
                }
                ,
              "schema": { "$ref": "#/definitions/AllJoynObject" }
            }
        }
      }
    }
  },
  "parameters": {
    "interface-ll" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : ["oic.if.ll"]
    },
    "interface-baseline" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : ["oic.if.baseline"]
    }
  },
  "definitions": {
    "AllJoynObject-ll" :
            {
        "allOf": [
          {
            "$ref": "../../core/schemas/oic.collection.linkslist-schema.json"
          }
        ]
      }

    ,
    "AllJoynObject" :
            {
        "allOf": [
          {
            "$ref": "../../core/schemas/oic.collection-schema.json"
          },
          {
            "properties": {
              "id": {
                "description": "Instance ID of this specific resource",
                "maxLength": 64,
                "readOnly": true,
                "type": "string"
              },
              "if": {
                "description": "The interface set supported by this resource",
                "items": {
                  "enum": [
                    "oic.if.baseline",
                    "oic.if.ll",
                    "oic.if.b",
                    "oic.if.lb",
                    "oic.if.rw",
                    "oic.if.r",
                    "oic.if.a",
                    "oic.if.s"
                  ],
                  "type": "string"
                },
                "minItems": 1,
                "readOnly": true,
                "type": "array"
              },
              "n": {
                "description": "Friendly name of the resource",
                "maxLength": 64,
                "readOnly": true,
                "type": "string"
              },
              "rt": {
                "items": {
                  "enum": [
                    "oic.r.alljoynobject",
                    "oic.wk.col"
                  ]
                },
                "maxItems": 2,
                "minItems": 2,
                "type": "array",
                "uniqueItems": true
              }
            }
          }
        ],
        "type": "object"
      }

  }
}
```

9.3.6 CRUDN behaviour

Example Resource URI	Create	Read	Update	Delete	Notify
/example/AllJoynObject/?if=oic.if.baseline		get	post		get
/example/AllJoynObject/?if=oic.if.ll		get			get