OCF Bridging Framework Specification

VERSION 2.2.2 | February 2021

CONTACT **admin@openconnectivity.org**

LEGAL DISCLAIMER

NOTHING CONTAINED IN THIS DOCUMENT SHALL BE DEEMED AS GRANTING YOU ANY KIND OF LICENSE IN ITS CONTENT, EITHER EXPRESSLY OR IMPLIEDLY, OR TO ANY INTELLECTUAL PROPERTY OWNED OR CONTROLLED BY ANY OF THE AUTHORS OR DEVELOPERS OF THIS DOCUMENT. THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "AS IS" BASIS, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE AUTHORS AND DEVELOPERS OF THIS SPECIFICATION HEREBY DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. OPEN CONNECTIVITY FOUNDATION, INC. FURTHER DISCLAIMS ANY AND ALL WARRANTIES OF NON-INFRINGEMENT, ACCURACY OR LACK OF VIRUSES.

CONTENTS

Figures

Tables

1 Scope

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

2 Normative references

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

AllJoyn About Interface Specification, *About Feature Interface Definitions*, Version 14.12
https://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
http://ieeexplore.ieee.org/servlet/opac?punumber=4610933

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

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

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

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

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

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

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

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

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

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

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

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

OCF Resource to Zigbee Cluster Mapping, *Open Connectivity Foundation Resource to Zigbee Cluster Mapping Specification*, version 2.0.3
Available at:
https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification_2.0.3.pdf
Latest version available at:
https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification.pdf

Zigbee, *Zigbee Specification*, August 2015
http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/

Zigbee Cluster Library, *Zigbee Cluster Library Specification*, January 2016
http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/

3 Terms, definitions, and abbreviated terms

3.1 Terms and definitions

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

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

– ISO Online browsing platform: available at https://www.iso.org/obp

– IEC Electropedia: available at http://www.electropedia.org/

3.1.1
Asymmetric Client Bridge

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

3.1.2
Asymmetric Server Bridge

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

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

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

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

3.1.6
Bridged Device
Bridged Client (3.1.3) or Bridged Server (3.1.8).

3.1.7
Bridged Protocol
another protocol (e.g., AllJoyn) that is being translated to or from OCF protocols

3.1.8
Bridged Resource
represents an artefact modelled and exposed by a Bridged Protocol (3.1.5), for example an AllJoyn object is a Bridged Resource.

3.1.9
Bridged Resource Type
schema used with a Bridged Protocol (3.1.5), for example AllJoyn Interfaces are Bridged Resource Types.

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

3.1.11
Bridging Function
Logic resident on the Bridge Platform (3.1.4) that performs that protocol mapping between OCF and the Bridged Protocol (3.1.7); a Bridge Platform (3.1.4) may contain multiple Bridging Functions dependent on the number of Bridged Protocols (3.1.7) supported.

3.1.12
OCF Bridge Device
OCF Device (3.1.11) that can represent devices that exist on the network but communicate using a Bridged Protocol (3.1.5) rather than OCF protocols.

3.1.13
OCF Client
logical entity that accesses an OCF Resource (3.1.12) on an OCF Server (3.1.15), which might be a Virtual OCF Server (3.1.26) exposed by the OCF Bridge Device (3.1.9)

3.1.14
OCF Device
logical entity that assumes one or more OCF roles (OCF Client (3.1.10), OCF Server (3.1.15). More than one OCF Device can exist on the same physical platform.

3.1.15
OCF Resource
represents an artefact modelled and exposed by the OCF Framework

3.1.16
OCF Resource Property
significant aspect or notion including metadata that is exposed through the OCF Resource (3.1.12)

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

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

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

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

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

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

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

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

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

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

3.1.27
Zigbee Attribute
data entity which represents a physical quantity or state within Zigbee. This data is communicated to other devices using commands.

3.1.28
Zigbee Cluster
one or more Zigbee Attributes (3.1.27), commands, behaviours, and dependencies, which supports an independent utility or application function. The term may also be used for an implementation or instance of such on an endpoint.

3.1.29
Zigbee Server
cluster interface which is listed in the input cluster list of the simple descriptor on an endpoint. Typically this interface supports all or most of the attributes of the cluster. A server cluster communicates with a corresponding remote client cluster with the same identifier.

3.1.30
Zigbee 3.0 Server
Zigbee Server (3.1.29) which is built on Zigbee 3.0 stack

3.1.31
Zigbee Client
cluster interface which is listed in the output cluster list of the simple descriptor on an endpoint. Typically this interface sends commands that manipulate the attributes on the corresponding Zigbee Server (3.1.29). A client cluster communicates with a corresponding remote server cluster with the same identifier.

3.1.32
Zigbee 3.0 Client
Zigbee Client (3.1.31) which is built on Zigbee 3.0 stack

3.1.33
Zigbee Device
unique device identifier and a set of mandatory and optional clusters to be implemented on a single Zigbee endpoint. The term may also be used for an implementation or instance on an endpoint. In this document, the unique identifier of a Zigbee Device maps to an OCF Device Type.

3.1.34
Zigbee 3.0 Device
Zigbee Device (3.1.33) which is built on Zigbee 3.0 stack

3.2 Abbreviated terms

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

3.2.2
CSV
Comma separated value

4 Document conventions and organization

4.1 Conventions

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

4.2 Notation

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

Required (or shall or mandatory).

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

Recommended (or should).

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

Allowed (or allowed).

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

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

Conditionally required (CR)

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

DEPRECATED

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

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

Words that are emphasized are printed in *italic*.

5 Introduction

5.1 Translation between OCF and Non-OCF ecosystem - primitive concept of Bridging

The details of Bridging may be implemented in many ways, for example, by using a Bridge Platform with an entity handler to interface directly to a Non-OCF device as shown in Figure 1.

Figure 1 – Server bridging to Non- OCF device

On start-up the Bridge Platform runs the entity handlers which discover the non-OCF systems (e.g., Heart Rate Sensor Device) and create Resources for each Device or functionality discovered. The entity handler creates a Resource for each discovered Device or functionality and binds itself to that Resource. These Resources are made discoverable by the Bridge Platform.

Once the Resources are created and made discoverable, then the Client Device can discover these Resources and operate on them using the mechanisms described in ISO/IEC 30118-1:2018. The requests to a Resource on the Bridge Platform are then interpreted by the entity handler and forwarded to the non-OCF device using the protocol supported by the non-OCF device. The returned information from the non-OCF device is then mapped to the appropriate response for that Resource.

Current OCF Bridging architecture implements the entity handler in the form of VOD.

5.2 Bridge Platform

This clause describes the functionality of a Bridge Platform; such a device is illustrated in Figure 2.

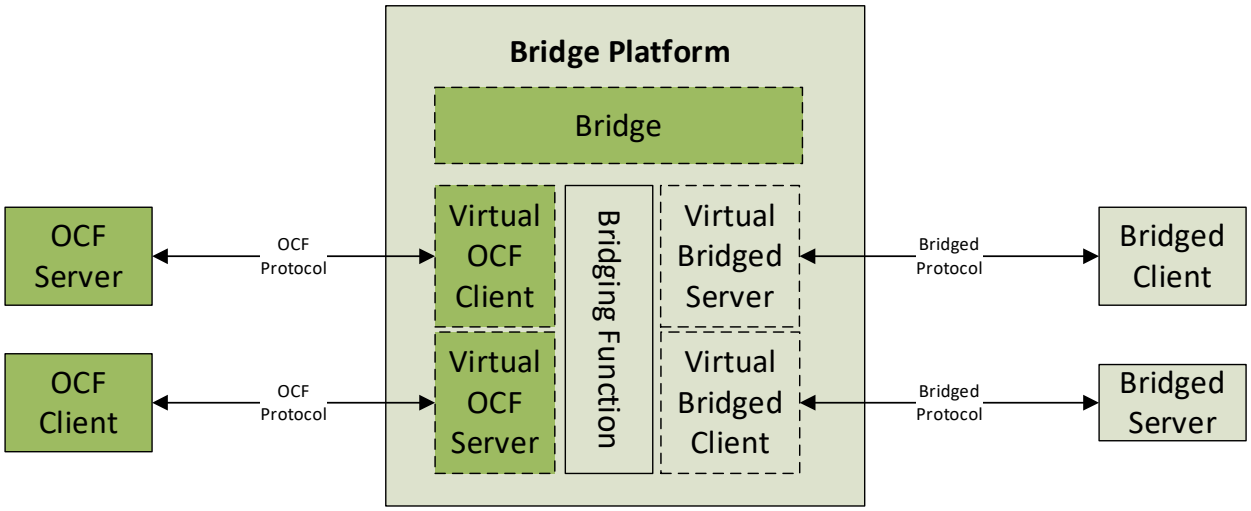

Figure 2 – Bridge Platform components

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

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

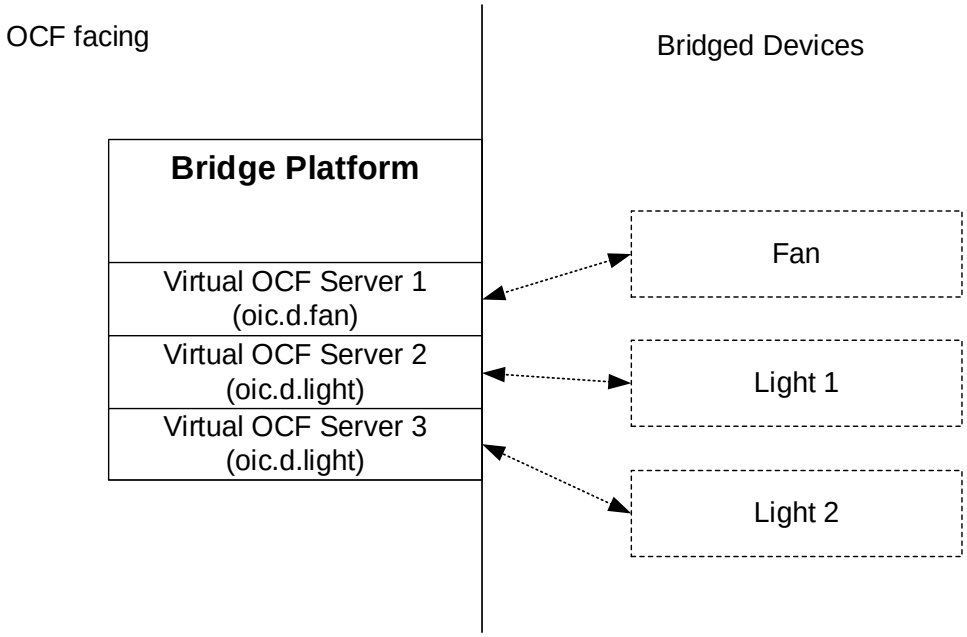

Figure 3 – Schematic overview of a Bridge Platform bridging non-OCF devices

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

5.3 Symmetric vs. asymmetric bridging

There are two kinds of bridging: Symmetric, Asymmetric. In symmetric bridging, a bridge device exposes OCF server(s) to another ecosystem and exposes other ecosystem's server(s) to OCF. In asymmetric bridging, a bridge device exposes OCF server(s) to another ecosystem or exposes another ecosystem's server(s) to OCF, but not both. The former case is called an Asymmetric Server Bridge (see Figure 4), the latter case is called an Asymmetric Client Bridge (see Figure 5)

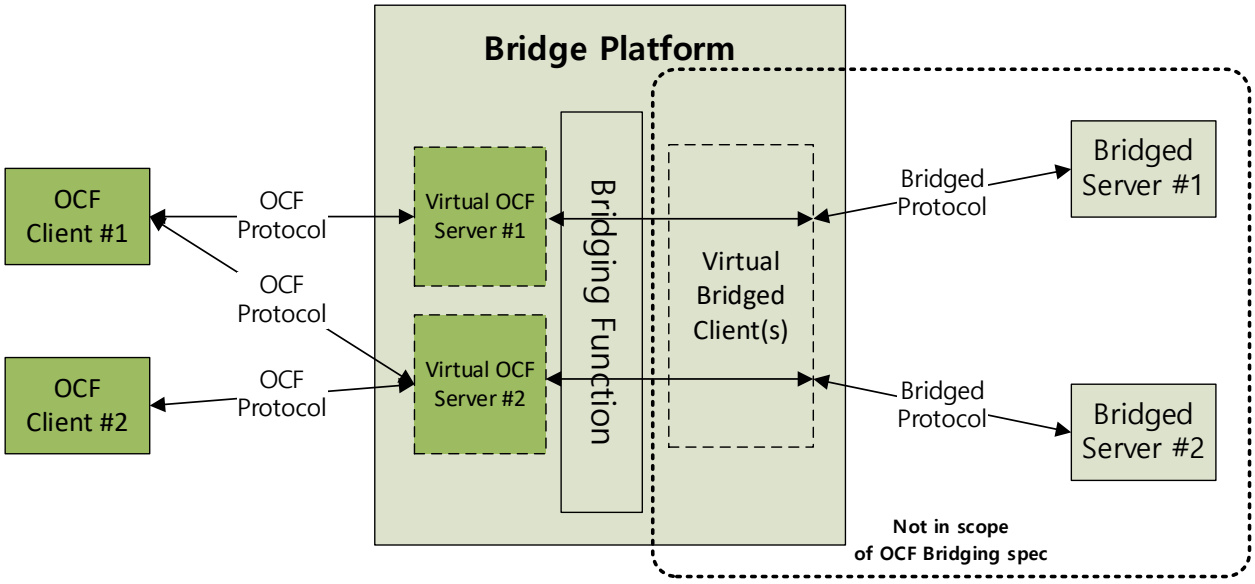

Figure 4 – Asymmetric server bridge

In Figure 4 each Bridged Server is exposed as a Virtual OCF Server to OCF side. These Virtual OCF Servers are same as normal OCF Servers except that they have additional rt value ("oic.d.virtual") for "/oic/d". The details of the Virtual Bridged Client are not in scope of this document.

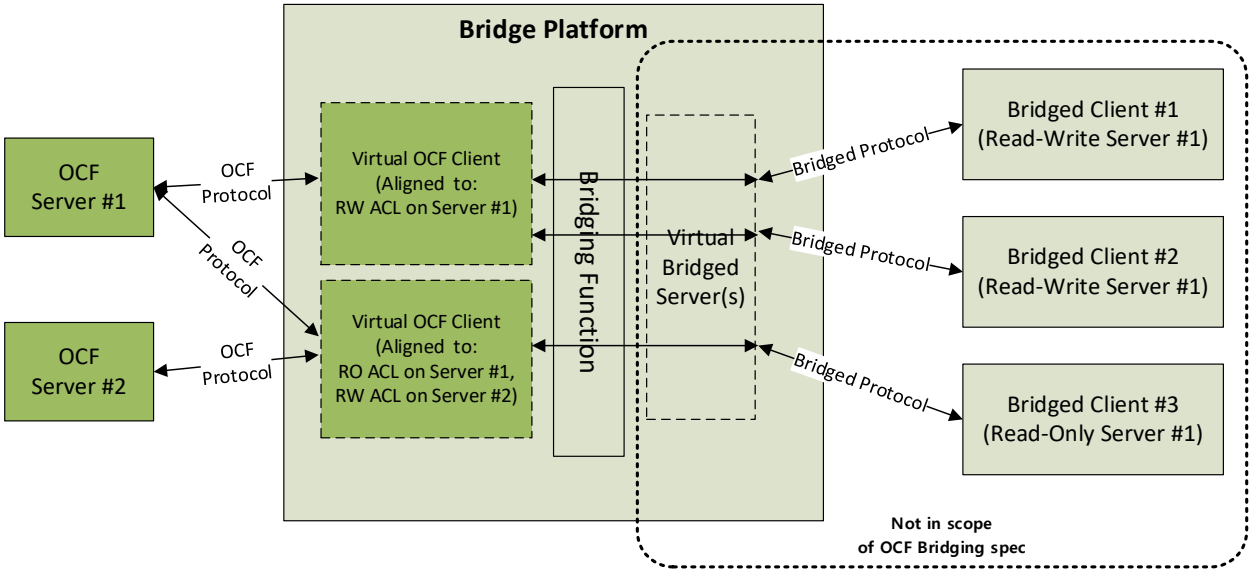

Figure 5 – Asymmetric client bridge

Figure 5 shows that each access to the OCF Server is modelled as a Virtual OCF Client. Those accesses can be aggregated if their target OCF servers and access permissions are same, therefore a Virtual OCF Client can tackle multiple Bridged Clients.

5.4 General requirements

5.4.1 Requirements common to all Bridge Platforms

A VOD shall have a Device Type that contains "oic.d.virtual". This allows Bridge Platforms to determine if a device is already being translated when multiple Bridge Platforms are present or Clients to determine if corresponding Server is a VOD or not.

Each Bridged Device shall be exposed as a separate Virtual OCF Server or Client, with its own OCF Endpoint, and set of mandatory Resources (as defined in ISO/IEC 30118-1:2018 and ISO/IEC 30118-2:2018).

Discovery of a VOD is the same as for an ordinary OCF Device; that is the VOD shall respond to multicast discovery requests. This allows platform-specific, device-specific, and resource-specific fields to all be preserved across translation.

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

5.4.2 Requirements specific to Symmetric Bridge Platforms

In addition to the requirements mentioned in 5.4.1, Symmetric Bridging shall satisfy following requirements.

The Bridge Platform shall check the protocol-independent UUID ("piid" in OCF) of each device and shall not advertise back into a Bridged Protocol a device originally seen via that Bridged Protocol. The Bridge Platform shall stop translating any Bridged Protocol device exposed in OCF via another Bridge Platform if the Bridge Platform sees the device via the Bridged Protocol. Similarly, the Bridge Platform shall not advertise an OCF Device back into OCF, and the Bridge Platform shall stop translating any OCF device exposed in the Bridged Protocol via another Bridge Platform if the Bridge Platform sees the device via OCF. These require that the Bridge Platform can determine when a device is already being translated. A VOD shall be indicated on the OCF Security Domain with a Device Type of "oic.d.virtual". How a Bridge Platform determines if a device is already being translated on a non-OCF Security Domain is described in the protocol-specific clauses (e.g. clause 1).

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

5.5 VOD List

For maintenance purposes, the Bridge maintains a list of VODs. This list includes Virtual OCF Servers and Virtual OCF Clients created by the Bridge Platform and subsequently on-boarded, as specified in ISO/IEC 30118-2:2018. A single instance of the Resource Type that defines the VOD list (see clause 7.2) shall be exposed by the Bridge. Please refer to ISO/IEC 30118-2:2018 for detailed operational requirements for the VOD list.

5.6 Resource discovery

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

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

For example, if a Bridge exposes VODs for the fan and lights shown in Figure 3, and an OCF Client performs a discovery request with a content format of "application/vnd.ocf+cbor", there will be four discrete responses, one for the Bridge, one for the virtual fan Device, and two for the virtual light Devices. Note that what is returned is not in the JSON format but in a suitable encoding as defined in ISO/IEC 30118-1:2018.

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



```
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/oic/sec/acl2",
    "rt": ["oic.r.acl2"],
    "if": ["oic.if.baseline"],
    "p": {"bm": 1},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/myIntrospection",
    "rt": ["oic.wk.introspection"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/myVodlist",
    "rt": ["oic.r.vodlist "],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
  }
 ]

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



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

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



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

Response from the second Light VOD:
[
  {
    "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
    "href": "/oic/res",
    "rt": ["oic.wk.res"],
    "if": ["oic.if.ll", "oic.if.baseline"],
```



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



```
    "rt": ["oic.wk.introspection"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
  }
]
```

Figure 6 – /oic/res example responses

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

When translating a service between a Bridged Protocol (e.g., AllJoyn) and OCF protocols, there are two possible types of translation. Bridge Platforms are expected to dedicate most of their logic to "deep translation" types of communication, in which data models used with the Bridged Protocol are mapped to the equivalent OCF Resource Types and vice-versa, in such a way that a compliant OCF Client or Bridged Client would be able to interact with the service without realising that a translation was made.

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

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

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

5.8 Security

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

6 Device type definitions

The required Resource Types are listed in Table 1.

Table 1 – Device type definitions

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

7 Resource type definitions

7.1 List of resource types

Table 2 lists the Resource Types defined in this document.

Table 2 – Alphabetical list of resource types

Friendly Name (informative)	Resource Type (rt)	Clause
VOD List	oic.r.vodlist	10.4

7.2 VOD List

7.2.1 Introduction

This Resource describes the VODs that have been onboarded on the Bridge Platform.

7.2.2 Example URI

/VODListResURI

7.2.3 Resource type

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

7.2.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "VOD List",
    "version": "2019-05-16",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "Copyright 2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": ["http"],
  "consumes": ["application/json"],
  "produces": ["application/json"],
  "paths": {
    "/VODListResURI" : {
      "get": {
        "description": "This Resource describes the VODs that have been onboarded on the Bridge
Platform.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"}
        ],
        "responses": {
          "200": {
            "description" : "Example response payload",
            "x-example":
            {
              "rt":    ["oic.r.vodlist"],
              "vods":  [
                {
                  "n": "Smoke sensor",
                  "di": "54919CA5-4101-4AE4-595B-353C51AA1234",
                  "econame": "Z-Wave"
                },
                {
                  "n": "Thermostat",
                  "di": "54919CA5-4101-4AE4-595B-353C51AA5678",
                  "econame": "Zigbee"
                }
              ]
            },
            "schema": { "$ref": "#/definitions/vodlist" }
          }
```



```
          }
        }
      }
    },
    "parameters": {
      "interface" : {
        "in" : "query",
        "name" : "if",
        "type" : "string",
        "enum" : ["oic.if.r", "oic.if.baseline"]
      }
    },
    "definitions": {
      "vodentry" : {
        "description": "Information for a VOD created by the Bridge",
        "type": "object",
        "properties": {
          "n": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
          },
          "di" : {
            "$ref": "https://openconnectivityfoundation.github.io/core/schemas/oic.types-
schema.json#/definitions/uuid"
          },
          "econame": {
            "description": "Ecosystem Name of the Bridged Device which is exposed by this VOD",
            "type": "string",
            "enum": [ "BLE", "oneM2M", "UPlus", "Zigbee", "Z-Wave" ],
            "readOnly": true
          }
        },
        "required": ["n", "di", "econame"]
      },
      "vodlist": {
        "type": "object",
        "properties": {
          "n": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
          },
          "rt" :  {
            "description": "Resource Type",
            "items": {
              "maxLength": 64,
              "type": "string",
              "enum": ["oic.r.vodlist"]
            },
            "minItems": 1,
            "uniqueItems": true,
            "readOnly": true,
            "type": "array"
          },
          "id": {
            "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
          },
          "if" :  {
            "description": "The OCF Interface set supported by this Resource",
            "items": {
              "enum": [
                "oic.if.baseline",
                "oic.if.r"
              ],
              "type": "string"
            },
            "minItems": 2,
```



```
            "uniqueItems": true,
            "readOnly": true,
            "type": "array"
          },
          "vods": {
            "description": "Array of information per VOD created by the Bridge",
            "type": "array",
            "minItems": 0,
            "uniqueItems": true,
            "readOnly": true,
            "items": {
              "$ref": "#/definitions/vodentry"
            }
          }
        },
        "required": ["vods"]
      }
    }
}
```

7.2.5 Property definition

Table 3 defines the Properties that are part of the "oic.r.vodlist" Resource Type.

Table 3 – The Property definitions of the Resource with type "rt" = "oic.r.vodlist".

Property name	Value type	Mandatory	Access mode	Description
if	array: see schema	No	Read Only	The OCF Interface set supported by this Resource
vods	array: see schema	Yes	Read Only	Array of information per VOD created by the Bridge
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type
econame	string	Yes	Read Only	Ecosystem Name of the Bridged Device which is exposed by this VOD
n	multiple types: see schema	Yes	Read Write	
di	multiple types: see schema	Yes	Read Write	

7.2.6 CRUDN behaviour

Table 4 defines the CRUDN operations that are supported on the "oic.r.vodlist" Resource Type.

Table 4 – The CRUDN operations of the Resource with type "rt" = "oic.r.vodlist".

Create	Read	Update	Delete	Notify
	get			observe

928

