OCF Security Specification

VERSION 2.2.7 | November 2023

OPEN CONNECTIVITY FOUNDATION™

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 INTERCONNECT CONSORTIUM, INC. FURTHER DISCLAIMS ANY AND ALL WARRANTIES OF NON-INFRINGEMENT, ACCURACY OR LACK OF VIRUSES.

The OCF logo is a trademark of Open Connectivity Foundation, Inc. in the United States or other countries. *Other names and brands may be claimed as the property of others.

CONTENTS

FIGURES

Tables

Introduction

This document, and all the other parts associated with this document, were developed in response to worldwide demand for smart home focused Internet of Things (IoT) devices, such as appliances, door locks, security cameras, sensors, and actuators; these to be modelled and securely controlled, locally and remotely, over an IP network.

While some inter-device communication existed, no universal language had been developed for the IoT. Device makers instead had to choose between disparate frameworks, limiting their market share, or developing across multiple ecosystems, increasing their costs. The burden then falls on end users to determine whether the products they want are compatible with the ecosystem they bought into, or find ways to integrate their devices into their network, and try to solve interoperability issues on their own.

In addition to the smart home, IoT deployments in commercial environments are hampered by a lack of security. This issue can be avoided by having a secure IoT communication framework, which this standard solves.

The goal of these documents is then to connect the next 25 billion devices for the IoT, providing secure and reliable device discovery and connectivity across multiple OSs and platforms. There are multiple proposals and forums driving different approaches, but no single solution addresses the majority of key requirements. This document and the associated parts enable industry consolidation around a common, secure, interoperable approach.

The OCF specification suite is made up of nineteen discrete documents, the documents fall into logical groupings as described herein:

- Core framework
  - Core Specification
  - Security Specification
  - Onboarding Tool Specification
- Bridging framework and bridges
  - Bridging Specification
  - Resource to Alljoyn Interface Mapping Specification
  - OCF Resource to oneM2M Resource Mapping Specification
  - OCF Resource to BLE Mapping Specification
  - OCF Resource to EnOcean Mapping Specification
  - OCF Resource to LWM2M Mapping Specification
  - OCF Resource to UPlus Mapping Specification
  - OCF Resource to Zigbee Cluster Mapping Specification
  - OCF Resource to Z-Wave Mapping Specification
- Resource and Device models
  - Resource Type Specification
  - Device Specification
- Core framework extensions
  - Easy Setup Specification
  - Core Optional Specification
- OCF Cloud
  - Cloud API for Cloud Services Specification

- Device to Cloud Services Specification
- Cloud Security Specification

OCF Security Specification

1 Scope

This document defines security objectives, philosophy, Resources and mechanism that impacts OCF base layers of ISO/IEC 30118-1. ISO/IEC 30118-1 contains informative security content. The OCF Security Specification contains security normative content and may contain informative content related to the OCF base or other OCF documents.

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.

ISO/IEC 30118-1 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 30118-3 Information technology -- Open Connectivity Foundation (OCF) Specification -- Part 3: Bridging specification
https://www.iso.org/standard/74240.html
Latest version available at:
https://openconnectivity.org/specs/OCF_Bridging_Specification.pdf

OCF Wi-Fi Easy Setup, Information technology – Open Connectivity Foundation (OCF) Specification – Part 7: Wi-Fi Easy Setup specification
Latest version available at:
https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf

OCF Cloud Specification, Information technology – Open Connectivity Foundation (OCF) Specification – Part 8: Cloud Specification
Latest version available at:
https://openconnectivity.org/specs/OCF_Cloud_Specification.pdf

OCF Cloud Security Specification - Open Connectivity Foundation (OCF) Specification – Cloud Security Specification
Latest version available at:

https://openconnectivity.org/specs/OCF_Cloud_Security_Specification.pdf

OCF Onboarding Tool Specification - Open Connectivity Foundation (OCF) Specification – Onboarding Tool Specification
Latest version available at:
https://openconnectivity.org/specs/OCF_Onboarding_Tool_Specification.pdf

OCF Cloud API for Cloud Services Specification - Open Connectivity Foundation (OCF) Cloud API for Cloud Services Specification
Latest version available at:

https://openconnectivity.org/specs/OCF_Cloud_API_For_Cloud_Services_Specification.pdf

JSON SCHEMA, draft version 4, http://json-schema.org/latest/json-schema-core.html.

IETF RFC 2315, *PKCS #7: Cryptographic Message Syntax Version 1.5*, March 1998,
https://tools.ietf.org/html/rfc2315

IETF RFC 2898, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, September 2000, https://tools.ietf.org/html/rfc2898

IETF RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7*, November 2000, https://tools.ietf.org/html/rfc2986

IETF RFC 4122, A Universally Unique IDentifier (UUID) URN Namespace, July 2005, https://tools.ietf.org/html/rfc4122

IETF RFC 4279, *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*, December 2005, https://tools.ietf.org/html/rfc4279

IETF RFC 4492, *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS),* May 2006, https://tools.ietf.org/html/rfc4492

IETF RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2,* August 2008, https://tools.ietf.org/html/rfc5246

IETF RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, May 2008, https://tools.ietf.org/html/rfc5280

IETF RFC 5489, *ECDHE_PSK Cipher Suites for Transport Layer Security (TLS),* March 2009, https://tools.ietf.org/html/rfc5489

IETF RFC 5545*, Internet Calendaring and Scheduling Core Object Specification (iCalendar),* September 2009, https://tools.ietf.org/html/rfc5545

IETF RFC 5755, *An Internet Attribute Certificate Profile for Authorization*, January 2010, https://tools.ietf.org/html/rfc5755

IETF RFC 6347, *Datagram Transport Layer Security Version 1.2*, January 2012, https://tools.ietf.org/html/rfc6347

IETF RFC 6655, *AES-CCM Cipher Suites for Transport Layer Security (TLS),* July 2012, https://tools.ietf.org/html/rfc6655

IETF RFC 7228, *Terminology for Constrained-Node Networks*, May 2014, https://tools.ietf.org/html/rfc7228

IETF RFC 7250, *Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, June 2014, https://tools.ietf.org/html/rfc7250

IETF RFC 7251, *AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS*, June 2014, https://tools.ietf.org/html/rfc7251

IETF RFC 7252, *The Constrained Application Protocol (CoAP)*, June 2014, https://tools.ietf.org/html/rfc7252

IETF RFC 8152, *CBOR Object Signing and Encryption (COSE)*, July 2017, https://tools.ietf.org/html/rfc8152

IETF RFC 8520, *Manufacturer Usage Description Specification*, Mar 2019, https://tools.ietf.org/html/rfc8520

IETF RFC 8613, *Object Security for Constrained RESTful Environments (OSCORE)*, July 2019, https://tools.ietf.org/html/rfc8613

oneM2M Release 3 Specifications, http://www.onem2m.org/technical/published-drafts

OpenAPI specification, aka *Swagger RESTful API Documentation Specification*, Version 2.0
https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md

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, ISO/IEC 30118-3 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
Access Management Service (AMS)
service that dynamically constructs ACL Resources in response to a Device Resource request

Note 1 to entry: An AMS can evaluate access policies remotely and supply the result to a Server which allows or denies a pending access request. An AMS is authorised to provision ACL Resources.

3.1.2
Credential Management Service (CMS)
Device that is authorized to provision credential Resources

3.1.3
Device Class
IETF RFC 7228 defined device class

3.1.4
Device Ownership Transfer Service (DOTS)
logical entity that establishes device ownership

3.1.5
End-Entity
any certificate holder which is not a Root or Intermediate Certificate Authority

Note 1 to entry: Typically, a device certificate.

3.1.6
Intermediary
Device that implements both Client and Server roles and may perform protocol translation, virtual device to physical device mapping or Resource translation

3.1.7
OCF Cipher Suite
set of algorithms and parameters that define the cryptographic functionality of a Device. The OCF Cipher Suite includes the definition of the public key group operations, signatures, and specific hashing and encoding used to support the public key.

3.1.8
OCF Rooted Certificate Chain
collection of X.509 v3 certificates in which each certificate chains to a trust anchor certificate which has been issued by a certificate authority under the direction, authority, and approval of the Open Connectivity Foundation Board of Directors as a trusted root for the OCF ecosystem.

3.1.9
Onboarding Tool (OBT)
tool that implements *DOTS*(3.1.4), *AMS*(3.1.1), and *CMS*(3.1.2) functionality

3.1.10
Out of Band Communication Channel
any mechanism for delivery of a secret from one party to another, not specified by OCF

3.1.11
Owner Credential (OC)
credential, provisioned to a Device, for the purposes of mutual authentication of the Device and *OBT*(3.1.9) during subsequent interactions, identified by having a Subject UUID matching the Resource Owner Id of the Device Ownership Transfer Resource hosted by a Device that has the credential

3.1.12
Role (Network context)
stereotyped behavior of a Device; one of [Client, Server or Intermediary]

3.1.13
Role Identifier
Property of an OCF credentials Resource or element in a role certificate that identifies a privileged role that a Server Device associates with a Client Device for the purposes of making authorization decisions when the Client Device requests access to Device Resources.

3.1.14
Secure Resource Manager (SRM)
module in the OCF Core that implements security functionality that includes management of security Resources such as ACLs, credentials and Device owner transfer state.

3.1.15
Security Virtual Resource (SVR)
Resource supporting security features.

Note 1 to entry: For a list of all the SVRs please see clause 13.

3.1.16
Trust Anchor
well-defined, shared authority, within a trust hierarchy, by which two cryptographic entities (e.g. a Device and an *OBT*(3.1.9)) can assume trust

3.1.17
Device Configuration Resource (DCR)
Resource that is any of the following:

a) a Discovery Core Resource, or
b) a Security Virtual Resource, or
c) a Wi-Fi Easy Setup Resource ("oic.r.easysetup", "oic.r.wificonf", "oic.r.devconf"), or
d) a CoAP Cloud Configuration Resource ("oic.r.coapcloudconf"), or
e) a Software Update Resource ("oic.r.softwareupdate"), or
f) a Maintenance Resource ("oic.wk.mnt").

3.1.18
Non-Configuration Resource (NCR)
Resource that is not a Device Configuration Resource (3.1.17)

3.1.19
OCF Security Domain
set of onboarded OCF Devices that are provisioned with credentialing information for confidential communication with one another

3.1.20
Owned (or "in Owned State")
having the "owned" Property of the "/oic/sec/doxm" Resource equal to "TRUE"

3.1.21
Unowned (or "in Unowned State")

having the "owned" Property of the "/oic/sec/doxm" Resource equal to "FALSE"

3.1.22
OCF Onboarding

initial establishment of ownership over a Device, and initial provisioning of the Device for normal operation

3.1.23
Auditable Event

system activity that may be indicative of a violation of security policy

3.1.24
Auditable Event Entry

record of the details of an Auditable Event

3.1.25
End User

person using the [particular] product

3.1.26
End-to-End Secure

securely encapsulate information so that *OCF Proxies* (3.1.28) on the end-to-end delivery path do not need to be trusted with the confidentiality, integrity and freshness of that information

3.1.27
End-to-End Security of Unicast Messages

interoperable mechanism which End-to-End Secures the exchange of unicast OCF CRUDN messages

3.1.28
OCF Proxy

functionality which can interpret the OCF compliant URIs of request messages intended for resources on another OCF Server and can route those request messages accordingly

3.1.29
Origin Client

Client which originally generated a request, as opposed to the Client functionality of a Proxy which is forwarding a request from another Device

3.1.30
OSCORE Master Secret

"Master Secret" as defined in clause 3.1 of IETF RFC 8613

3.1.31
OSCORE Recipient ID

"Recipient ID" as defined in clause 3.1 of IETF RFC 8613

3.1.32
OSCORE Security Context

"Security Context" as defined in clause 3.1 of IETF RFC 8613

3.1.33
OSCORE Sender ID

"Sender ID" as defined in clause 3.1 of IETF RFC 8613

3.1.34
OSCORE Sender Sequence Number

"Sender Sequence Number" as defined in clause 3.1 of IETF RFC 8613

3.1.35
Target Server

Server to which a request is addressed, as opposed to the Server functionality of a *OCF Proxy* (3.1.28) which receives a request to be forwarded to another Device

3.1.36
Simple Secure Multicast

delivery of UPDATE request messages from a Client to a group of Servers using network-layer multicast, where the messages are protected with a simple security mechanism

3.1.37
Simple Secure Multicast Client Context

OSCORE Security Context (3.1.32) parameters provisioned to the Client of a *Simple Secure Multicast Group* (3.1.38) to enable End-to-End Security of *Simple Secure Multicast Requests* (3.1.39) sent to Servers of that *Simple Secure Multicast Group* (3.1.38)

3.1.38
Simple Secure Multicast Group

group of Servers and one (1) associated Client provisioned with credentials to enable *Simple Secure Multicast* (3.1.36) from the Client to the set of Servers

3.1.39
Simple Secure Multicast Request

OSCORE-protected UPDATE request message delivered from a Client to a group of Servers using *Simple Secure Multicast* (3.1.36)

3.1.40
Simple Secure Multicast Server Context

OSCORE Security Context parameters provisioned to Servers of a Simple Secure Multicast Group (3.1.38) to enable End-to-End Security of *Simple Secure Multicast Requests* (3.1.39) sent by the Client of that *Simple Secure Multicast Group* (3.1.38)

3.1.41
Device Onboarding Connection (DOC)

special DTLS connection established for the purposes of onboarding the Device securely when a Device is in RFOTM

NOTE: The Owner Transfer Method selected will determine the specifics of the DOC used.

3.1.42
Ready For Normal Operation State

state of a Device in which *NCRs* (3.1.18) can be accessed

3.1.43
Ready For Owner Transfer Mechanism State

state of a Device in which a Device can be Onboarded

3.1.44
Ready For Provisioning State

state of a Device in which *SVRs* (3.1.15) can be configured

3.1.45
Reset State

state of a Device in which the configurable Properties of Device's resources are reset to the manufacturer default and the Device becomes *Unowned* (3.1.21)

3.1.46
Soft Reset State

state of a Device in which *SVRs* (3.1.15) can be configured, with slightly more Properties available than in RFPRO

3.2 Symbols and abbreviated terms

AC	Access Control
ACE	Access Control Entry
ACL	Access Control List
AEAD	Authenticated Encryption with Authenticated Data

NOTE: Defined in IETF RFC 8152

AEE	Auditable Event Entry
AES	Advanced Encryption Standard
AMS	Access Management Service
CMS	Credential Management Service
COSE	CBOR Object Signing and Encryption

NOTE: Defined in IETF RFC 8152

CRUDN	CREATE, RETREIVE, UPDATE, DELETE, NOTIFY
CSR	Certificate Signing Request
DOC	Device Onboarding Connection
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EKU	Extended Key Usage
DOTS	Device Ownership Transfer Service
ID	Identity/Identifier
JSON	JavaScript Object Notation.
NVRAM	Non-Volatile Random-Access Memory
OC	Owner Credential
OCSP	Online Certificate Status Protocol
OBT	Onboarding Tool
OID	Object Identifier

OSCORE	Object Security for Constrained RESTful Environments

NOTE: Defined in IETF RFC 8613

OTM	Owner Transfer Method
PE	Policy Engine
PIN	Personal Identification Number
PPSK	PIN-authenticated pre-shared key
PRF	Pseudo Random Function
PSI	Persistent Storage Interface
PSK	Pre Shared Key
RBAC	Role Based Access Control
RM	Resource Manager
RNG	Random Number Generator
RESET	Reset State
RFNOP	Ready For Normal Operation State
RFOTM	Ready For Owner Transfer Mechanism State
RFPRO	Ready For Provisioning State
SBAC	Subject Based Access Control
SEE	Secure Execution Environment
SRESET	Soft Reset State
SRM	Secure Resource Manager
SSM	Simple Secure Multicast
SVR	Security Virtual Resource
URI	Uniform Resource Identifier
VOD	Virtual OCF Device

4 Document conventions and organization

4.1 Conventions

This document defines Resources, protocols and conventions used to implement security for OCF core framework and applications.

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

In this document, to be consistent with the IETF usages for RESTful operations, the RESTful operation words CRUDN, CREATE, RETRIVE, UPDATE, DELETE, and NOTIFY will have all letters capitalized. Any lowercase uses of these words have the normal technical English meaning.

Figure 1 depicts interaction between OCF Devices.

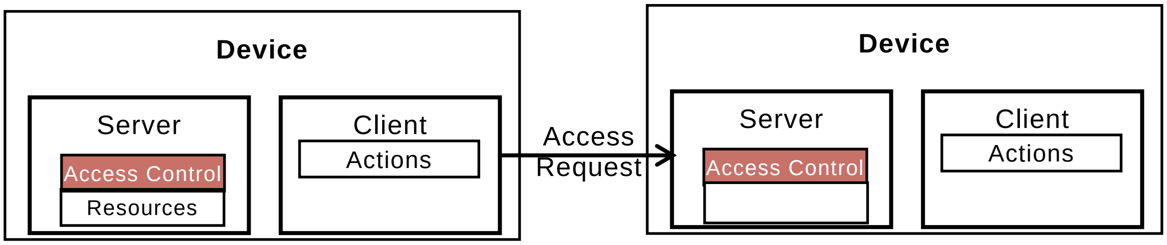


Figure 1 – OCF interaction

Devices may implement a Client role that performs Actions on Servers. Actions access Resources managed by Servers. The OCF stack enforces access policies on Resources. End-to-end Device interaction can be protected using session protection protocol (e.g. DTLS) or with data encryption methods.

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 OCF 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 OCF Core Architecture and should be implemented. Recommended features take advantage of the capabilities OCF 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 (may or allowed).

These features are neither required nor recommended by OCF 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.

4.3 Data types

See ISO/IEC 30118-1.

4.4 Document structure

Informative clauses may be found in the Overview clauses, while normative clauses fall outside of those clauses.

The Security Specification may use the OpenAPI specification as the API definition language. The mapping of the CRUDN actions is specified in ISO/IEC 30118-1.

5 Security overview

5.1 Security model of operation

The goal of OCF's security architecture is to protect the data and device states represented by the OCF Resources. From the OCF perspective, a Device is a certifiable logical entity that participates in an OCF ecosystem. During interactions between Devices, the Device acting as the Server holds and controls the Resources and provides the Device acting as a Client access to those Resources, subject to a set of security mechanisms and conforming to the policies configured by the OCF Security Domain Owner. The Platform hosting the Device may provide security hardening to ensure robustness of the variety of operations described in this document. Multiple Devices may be hosted by the same Platform.

The security model of operation for direct Device-to-Device interaction (that is, exchanges which are not facilitated by entities acting as OCF Proxies between the Client and Server) is depicted in Figure 2 and described in the following steps:

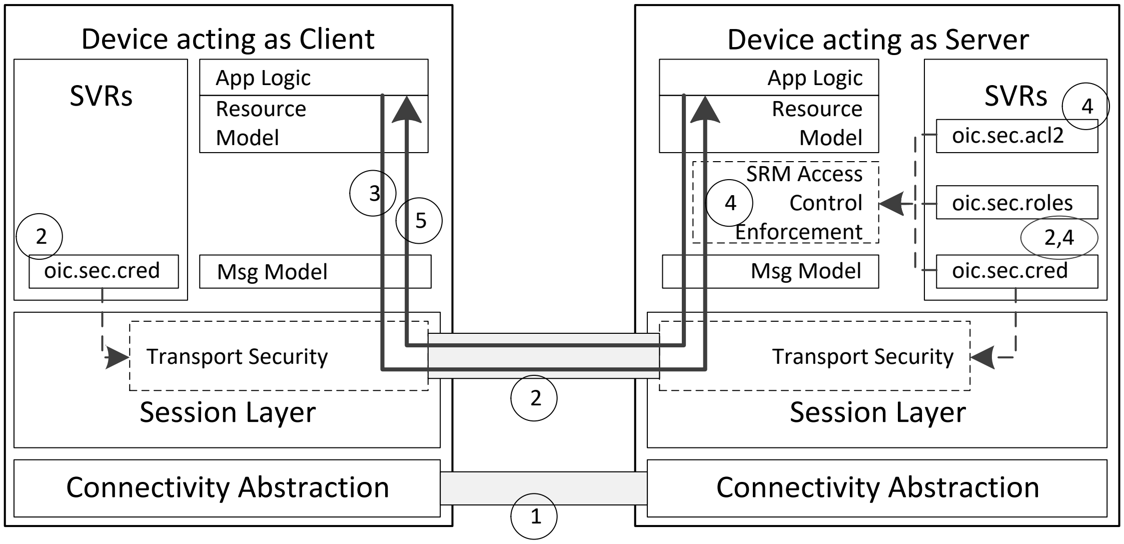


Figure 2 – OCF layers for direct Device-to-Device interaction

1) The Client establishes a network connection to the Server (Device holding the Resources).

2) The Devices (Server and Client) exchange messages either via a mutually-authenticated secure channel between the two Devices or via an unsecure connection.

   a) The "/oic/sec/cred" Resource on each Device holds the credentials used for mutual authentication and credentials used for role authorization.

   b) Messages received over a secured channel are associated with a "deviceUUID". In the case of a certificate credential, the "deviceUUID" is part of the certificate received from the other Device. In the case of a symmetric key credential, the "deviceUUID" is associated with the credential in the "/oic/sec/cred" Resource.

   c) The Client may present its role certificate to request association with a role identifier ("roleid").The Server may associate the Client with any number of role identifiers.

   d) Requests received by a Server over an unsecured channel are treated as anonymous and are not associated with any "deviceUUID" or "roleid".

3) The Client submits a request to the Server.

4) The Server receives the request.

   a) If the request is received over an unsecured channel, the Server treats the request as anonymous and no "deviceUUID" or "roleid" are associated with the request.

   b) If the request is received over a secured channel, then the Server associates the request with the "deviceUUID" of the Client and all valid "roleid" values of the Client by default.

   c) The Server then consults the Access Control List (ACL), and looks for an Access Control Entry (ACE) matching the following criteria:

      i) The requested Resource matches a Resource reference in the ACE

      ii) The requested operation is permitted by the "permissions" of the ACE, and

      iii) The "subjectUUID" contains either one of a special set of wildcard values or, if the Device is not anonymous, the subject matches the Client "deviceUUID" associated with the request or a valid "roleid" associated with the request. The special wildcard values authorize all Devices communicating over either authenticated and encrypted sessions or unsecured sessions to interact according to the ACE.

      If there is a matching ACE, then access to the Resource is permitted; otherwise access is denied. Access is enforced by the Server’s Secure Resource Manager (SRM).

5) The Server sends a response back to the Client.

OCF also supports exchange of messages between an Origin Client and Target Server facilitated at one or more entities acting as OCF Proxies.

NOTE 1: Any number of OCF Proxies may be on the path between the Origin Client and Target Server, although this number is expected to be small in practice.

In some scenarios, an OCF Proxy acts as a Server to incoming OCF CRUDN request messages: processing the OCF CRUDN request messages; and then sending appropriate OCF CRUDN request messages onwards towards the Target Server. The OCF Proxy can also process the corresponding incoming OCF CRUDN response message and send appropriate OCF CRUDN request messages back towards the Origin Client.

This approach implies that the owner of the Security Domain (containing the Origin Client and Target Server) is willing to trust all OCF Proxies on the message delivery path with the confidentiality, integrity and freshness of the OCF CRUDN messages. Alternatively, the Origin Client and Target Server can apply End-to-End Security of Unicast Messages which enables securing the exchange of OCF CRUDN messages so that OCF Proxies do not need to be trusted with the confidentiality and integrity of the OCF CRUDN messages.

The security model of operation when using OCF Proxies without End-to-End Security of Unicast Messages is described in OCF Cloud Specification, OCF Cloud Security Specification, and C2C API.

Figure 3 and Figure 4 depict the security model of operation when using OCF Proxies and End-to-End Security of Messages is applied; see also the following steps. Figure 3 illustrates an example with one OCF Proxy. Figure 4 illustrates a more complex example with two OCF Proxies using OCF Cloud API for Cloud Services Specification; see notes 1 and 2.

NOTE 2: If the OCF Proxies in Figure 4 are OCF Clouds, OCF Proxy A is the Origin Cloud to which the Origin Client is registered, and OCF Proxy B is the Target Cloud to which the Target Server is registered.

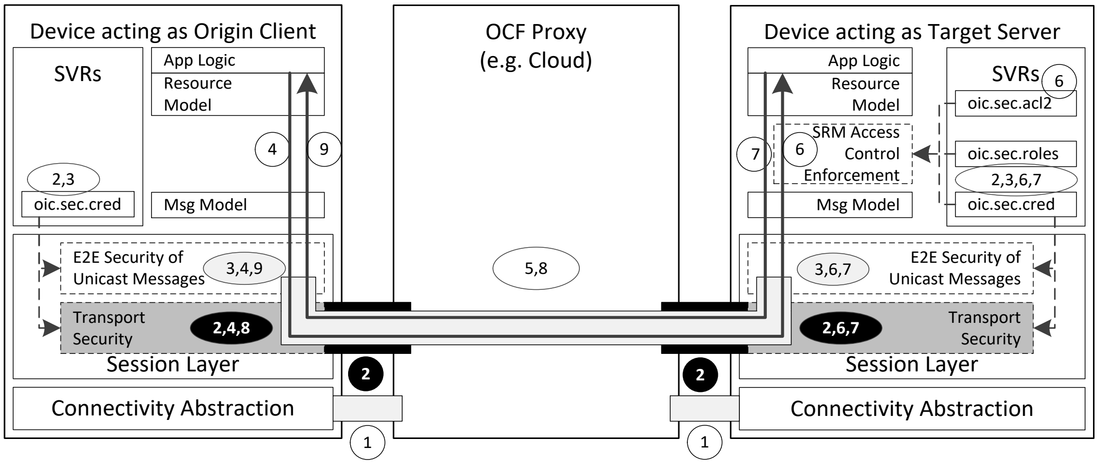


Figure 3 – OCF layers for interactions via one OCF Proxy

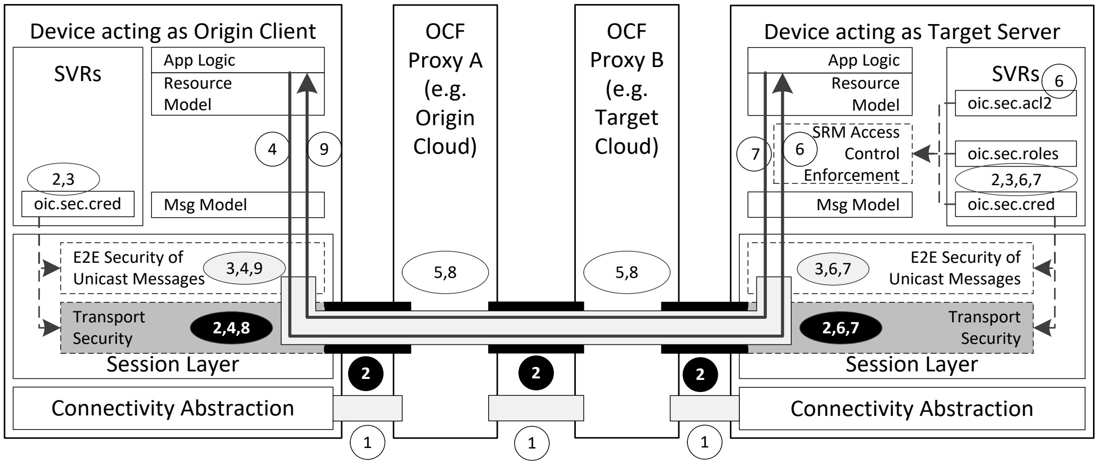


Figure 4 – OCF layers for interactions via two OCF Proxies

1) Pairwise network connections are established.

2) Messages are exchanged over each network connection via pairwise mutually-authenticated secure transport connection.

3) The Origin Client and Target Server establish an End-to-End Secured channel which is mutually-authenticated using credentials held in the "/oic/sec/cred" Resources of the Origin Client and Target Server.

4) The Origin Client generates an OCF CRUDN request message to the Target Server. The Origin Client encapsulates the OCF CRUDN request message into an End-to-End Secured request message of the End-to-End Secured channel (established in step 3). Information identifying the Target Server is left un-encrypted in the End-to-End Secured request message, so OCF Proxies can use the identifying information to route the End-to-End Secured request message correctly. The Origin Client sends the End-to-End Secured request message to its OCF Proxy, over the optionally secured transport connection established with that OCF Proxy. See Note 3.

5) Each OCF Proxy on the path extracts the identifying information of the Target Server from the request message and, subject to the OCF Proxy's policies governing End-to-End Secured request messages, forwards the end-to- End-to-End Secured request message towards the Target Server over an optionally secured transport connection. See notes 3, 4 and 5.

6) The Target Server verifies and decrypts the End-to-End Secured request message as a message of the End-to-End Secured channel (established at step 3) to extract the encapsulated OCF CRUDN request message from the Origin Client. The OCF CRUDN request message is treated as being received over an authenticated encrypted ("auth-crypt") connection and associated with a "deviceUUID". The "deviceUUID" is associated with the credential in the "/oic/sec/cred" Resource used to establish the End-to-End Secured channel in step 3.

7) The Target Server determines whether access to the resource is permitted as described in step 4c of the Security model for direct Device-to-Device interaction shown in Figure 2.

8) The Target Server generates an OCF CRUDN response message and encapsulates the OCF CRUDN response message into an End-to-End Secured response message of the End-to-End Secured channel (established at step 3). The Target Secure sends the End-to-End Secured response message to its OCF Proxy, over the optionally secured transport connection on which the corresponding request was received. See Note 3.

9) Each OCF Proxy on the path forwards the End-to-End Secured response message towards the Origin Client over the optionally secured transport connection on which the corresponding request message was received. See Note 3.

10) The Origin Client verifies and decrypts the End-to-End Secured response message as a message of the End-to-End Secured channel (established at step 3) to extract the encapsulated OCF CRUDN response message from the Target Server.

NOTE 3: While in transit, the OCF CRUDN message might be secured by up to two independent layers of Security: a layer of End-to-End Security of Unicast Messages (using OSCORE), and an independent layer of transport Security (using DTLS or TLS).

NOTE 4: This document does not address details of how an OCF Proxy determines if its policies permit forwarding the request message towards the identified Target Server. If an OCF Proxy permits forwarding a request message towards a Target Server, then it is assumed that the OCF Proxy also permits forwarding the corresponding response message(s) over the transport connection on which the corresponding request message was received.

NOTE 5: This document does not address how OCF Proxy A determines that OCF Proxy B is the correct OCF Proxy to forward the request message to. The OCF Cloud API for Cloud Services Specification provides the details for the case where the OCF Proxy A and OCF Proxy B are OCF Clouds.

As shown in Figure 5, Simple Secure Multicast (SSM) enables a Client to securely communicate an UPDATE request to a group of Servers with a single non-confirmable UPDATE request delivered via networking-layer multicast.

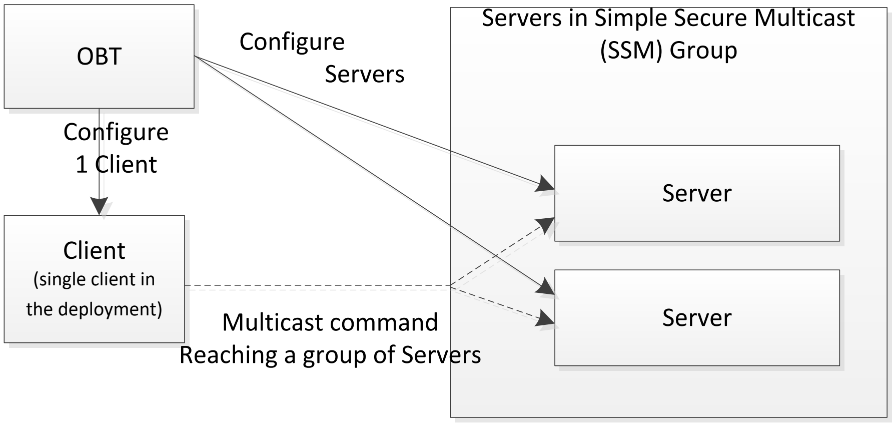


Figure 5 – Single request reaches a group of Servers

The Security model for SSM is described in Figure 6 and the accompanying steps.

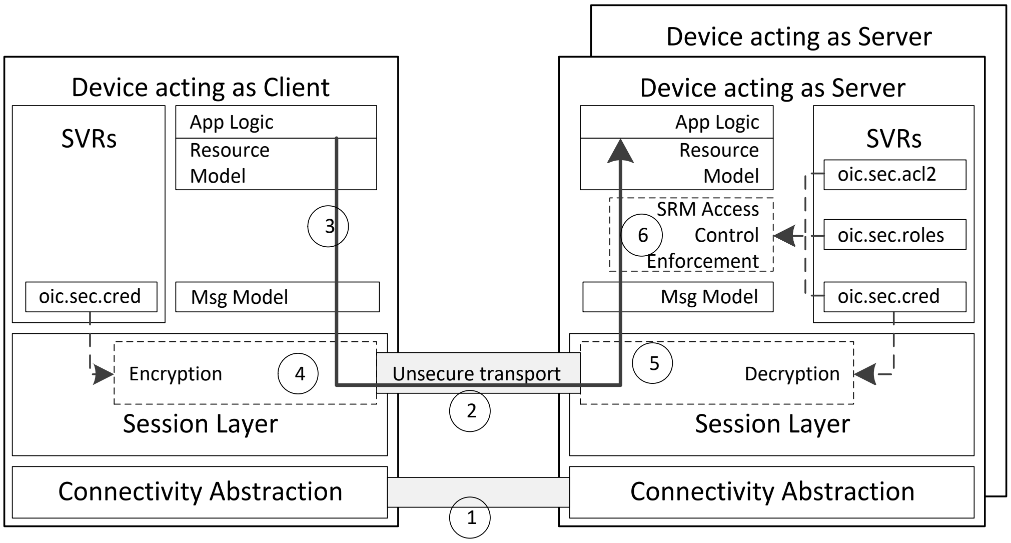


Figure 6 – OCF Layers for Simple Secure Multicast

1) The Client and Servers in the SSM Group are configured with encryption/decryption. The Client knows the preconfigured multicast address to use and how to create the actual payload of the command to send.

2) Messages are exchanged over an unsecure transport connection.

3) The Client generates an UPDATE request message to the Servers.

4) The Client encapsulates the UPDATE request message into an End-to-End Secured request message of the unsecured channel. The multicast address is left unencrypted in the Secured request message.

The Client sends the Secured UPDATE request message to the multicast URL of the Servers, using the URL of the multicast enabled resource.

5) The Servers decrypt the message. The UPDATE request message is treated as being received over an authenticated encrypted ("auth-crypt") connection and associated with a "deviceUUID" (which can be the Device UUID of the Client).

6) The Server determines whether access to the Resource is permitted as described in step 4c of the Security model for direct Device-to-Device interaction shown in Figure 2.

Resource protection includes protection of data both while at rest and during transit. Aside from access control mechanisms, the OCF Security Specification does not include specification of secure storage of Resources. Secure storage may be accomplished through the use of hardware security or encryption of data at rest. The exact implementation of secure storage is subject to a set of hardening requirements that are specified in clause 14 and may be subject to certification guidelines.

Data in transit protection is specified fully as a normative part of this document. This document supports data in transit data protection at the transport layer through use of mechanisms such as DTLS and end-to-end data-in-transit protection through OSCORE.

NOTE 6: DTLS will provide packet by packet protection, rather than protection for the OCF CRUDN message as whole. For instance, if the integrity of the entire OCF CRUDN message as a whole is required, separate end-to-end Security (for example, using OSCORE) should be applied before passing the packet down to the transport layer.

Figure 7 depicts OCF Security Enforcement Points.

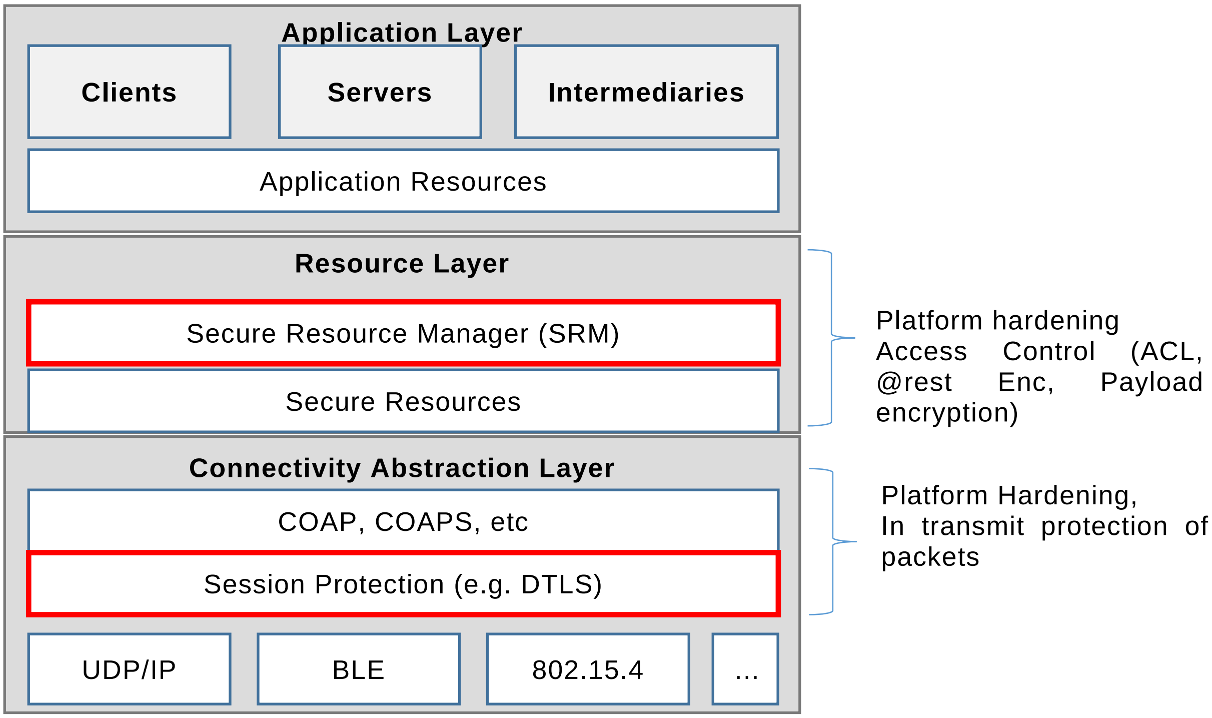

Application Layer
Clients
Servers
Intermediaries
Application Resources
Resource Layer
Secure Resource Manager (SRM)
Secure Resources
Platform hardening
Access Control (ACL, @rest Enc, Payload encryption)
Connectivity Abstraction Layer
COAP, COAPS, etc
Session Protection (e.g. DTLS)
Platform Hardening,
In transmit protection of packets
UDP/IP
BLE
802.15.4
…

Figure 7 – OCF security enforcement points

5.2 Access control

5.2.1 Access control general

The OCF framework assumes that Resources are hosted by a Server and are made available to Clients subject to access control and authorization mechanisms. The Resources at the Server are protected through implementation of access control, authentication and confidentiality protection.

This clause provides an overview of access control through the use of Access Control Lists. However, access control in OCF is agnostic regarding transport and connectivity abstraction layers.

Implementation of access control relies on a-priori definition of a set of access policies for the Resource. The policies are stored locally in an ACL Resource provisioned by an Access Management Service (AMS) in the form of Access Control Entries (ACE). The lack of such an associated ACE results in the Resource being inaccessible. Multiple types of access control mechanisms may be applied:

– Subject-based access control (SBAC), where the ACE matches the identity of the Client against the subject included in the policy defined for the Resource. Asserting the identity of the Client requires an authentication process.

– Role-based Access Control (RBAC), where the ACE matches a role identifier included in the policy for the Resource to a role identifier associated with the Client.

– Wildcard-based Access Control, where the ACE matches a connection type, used to access the Resource (i.e. any mutually-authenticated connection).

The ACE only applies if the ACE matches both the subject (i.e. Client) and the requested Resource. There are multiple ways a subject could be matched, (1) Device UUID, (2) Role Identifier or (3) wildcard. The way in which the Client connects to the Server may be relevant for making access control decisions. Wildcard matching on authenticated vs. unauthenticated and encrypted vs. unencrypted connection allows an access policy to be broadly applied to subject classes.

Example Wildcard Matching Policy:

```
"aclist2": [
 {
  "subject": {"conntype" : "anon-clear" },
  "resources":[
    { "wc":"*" }
  ],
  "permission": 31
  },
  {
   "subject": {"conntype" : "auth-crypt" },
   "resources":[
    { "wc":"*" }
  ],
  "permission": 31
  },
]
```

Details of the format for ACL are defined in clause 12. The ACL is composed of one or more ACEs.

Some Resources, such as Collections, generate requests to linked Resources when appropriate Interfaces are used. In such cases, additional access control considerations are necessary. Additional access control considerations for Collections when using the batch OCF Interface are found in clause 12.2.7.3. ACL Resource requires the same security protection as other sensitive Resources when it comes to both storage and handling by the SRM.

5.2.2 ACL architecture

The Server examines the Resource(s) requested by the client before processing the request. The access control Resource is searched to find one or more ACE entries that match the Client and the requested Resources. If a match is found, then permission and period constraints are applied. If more than one match is found, then each ACE entry is evaluated for a match independently.

The Server uses the connection context to determine whether the subject has authenticated or not and whether data confidentiality has been applied or not. If the user has authenticated, then subject matching may happen at increased granularity based on role or device identity.

Each ACE contains the permission set that will be applied for a given Client. Permissions consist of a combination of CREATE, RETREIVE, UPDATE, DELETE and NOTIFY (CRUDN) actions. Clients authenticate as a Device and optionally operating with one or more roles. Devices may acquire elevated access permissions when asserting a role. For example, an "oic.role.owner" role might expose additional Resources and OCF Interfaces not normally accessible.

Servers host ACL Resources locally. Local ACLs allow greater autonomy in access control processing.
The following use cases describe the operation of access control:

Use Case 1: As depicted in Figure 8, Server Device hosts 4 Resources (R1, R2, R3 and R4). Client Device D1 requests access to Resource R1 hosted at Server Device 5. ACL[0] corresponds to Resource R1 and includes D1 as an authorized subject. Thus, Device D1 receives access to Resource R1 because the local ACL "/oic/sec/acl2/0" matches the request.

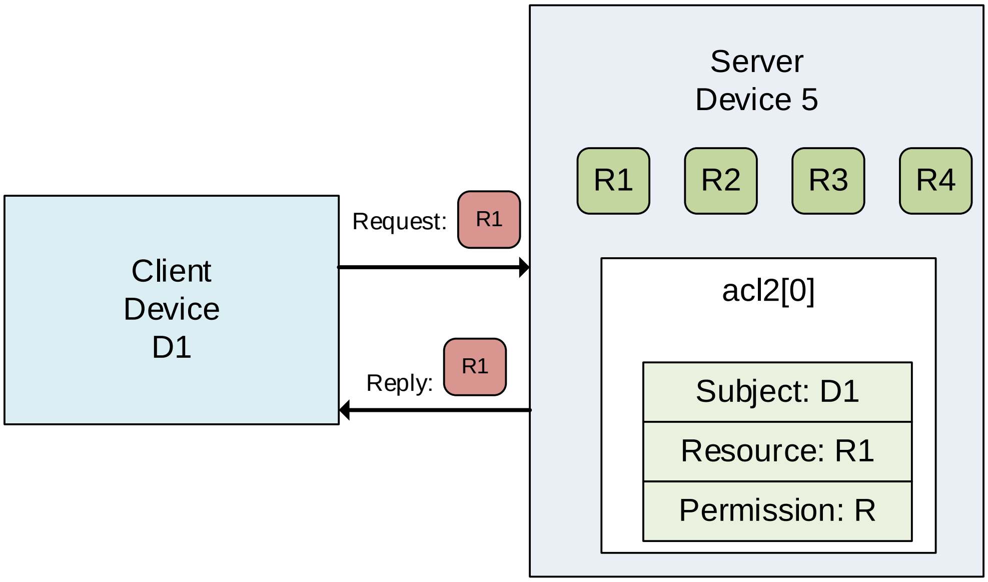


Figure 8 – Use case-1 showing simple ACL enforcement

5.3 Onboarding overview

5.3.1 Onboarding general

Before a Device becomes operational in an OCF environment and is able to interact with other Devices, it needs to be appropriately onboarded. The first step in onboarding a Device is to configure the ownership where the legitimate user that owns/purchases the Device uses an Onboarding tool (OBT) and using the OBT uses one of the Owner Transfer Methods (OTMs) to establish ownership. Once ownership is established, the OBT provisions the Device, at the end of which the Device becomes operational and is able to interact with other Devices in an OCF environment.

Figure 9 depicts an overview of Onboarding.

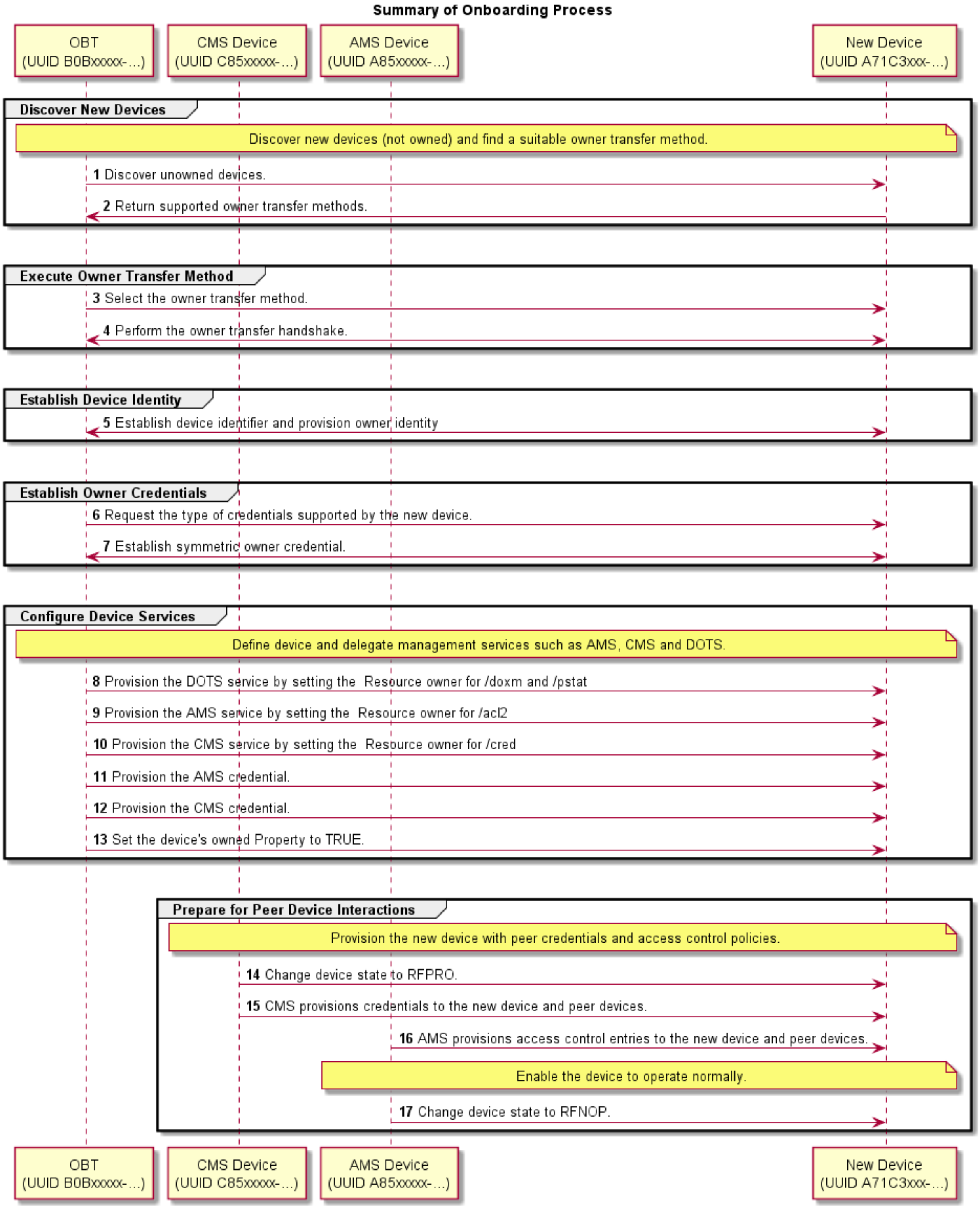

Summary of Onboarding Process
OBT (UUID B0Bxxxxx-...)
CMS Device (UUID C85xxxxx-...)
AMS Device (UUID A85xxxxx-...)
New Device (UUID A71C3xxx-...)
Discover New Devices
Discover new devices (not owned) and find a suitable owner transfer method.
1 Discover unowned devices.
2 Return supported owner transfer methods.
Execute Owner Transfer Method
3 Select the owner transfer method.
4 Perform the owner transfer handshake.
Establish Device Identity
5 Establish device identifier and provision owner identity
Establish Owner Credentials
6 Request the type of credentials supported by the new device.
7 Establish symmetric owner credential.
Configure Device Services
Define device and delegate management services such as AMS, CMS and DOTS.
8 Provision the DOTS service by setting the Resource owner for /doxm and /pstat
9 Provision the AMS service by setting the Resource owner for /acl2
10 Provision the CMS service by setting the Resource owner for /cred
11 Provision the AMS credential.
12 Provision the CMS credential.
13 Set the device's owned Property to TRUE.
Prepare for Peer Device Interactions
Provision the new device with peer credentials and access control policies.
14 Change device state to RFPRO.
15 CMS provisions credentials to the new device and peer devices.
16 AMS provisions access control entries to the new device and peer devices.
Enable the device to operate normally.
17 Change device state to RFNOP.
OBT (UUID B0Bxxxxx-...)
CMS Device (UUID C85xxxxx-...)
AMS Device (UUID A85xxxxx-...)
New Device (UUID A71C3xxx-...)

Figure 9 – Onboarding overview

This clause explains the onboarding and security provisioning process but leaves the provisioning of non-security aspects to other OCF documents. In the context of security, all Devices are required to be provisioned with minimal security configuration that allows the Device to securely interact/communicate with other Devices in an OCF environment. This minimal security configuration is defined as the Onboarded Device RFNOP and is specified in 8.

5.3.2 Onboarding steps

The flowchart in Figure 10 shows the typical steps that are involved during onboarding. Although onboarding may include a variety of non-security related steps, the diagram focus is mainly on the security related configuration to allow a new Device to function within an OCF environment. Onboarding typically begins with the Device becoming an Owned Device followed by configuring the Device for the environment that it will operate in. This would include setting information such as who may access the Device and what actions may be performed as well as what permissions the Device has for interacting with other Devices.

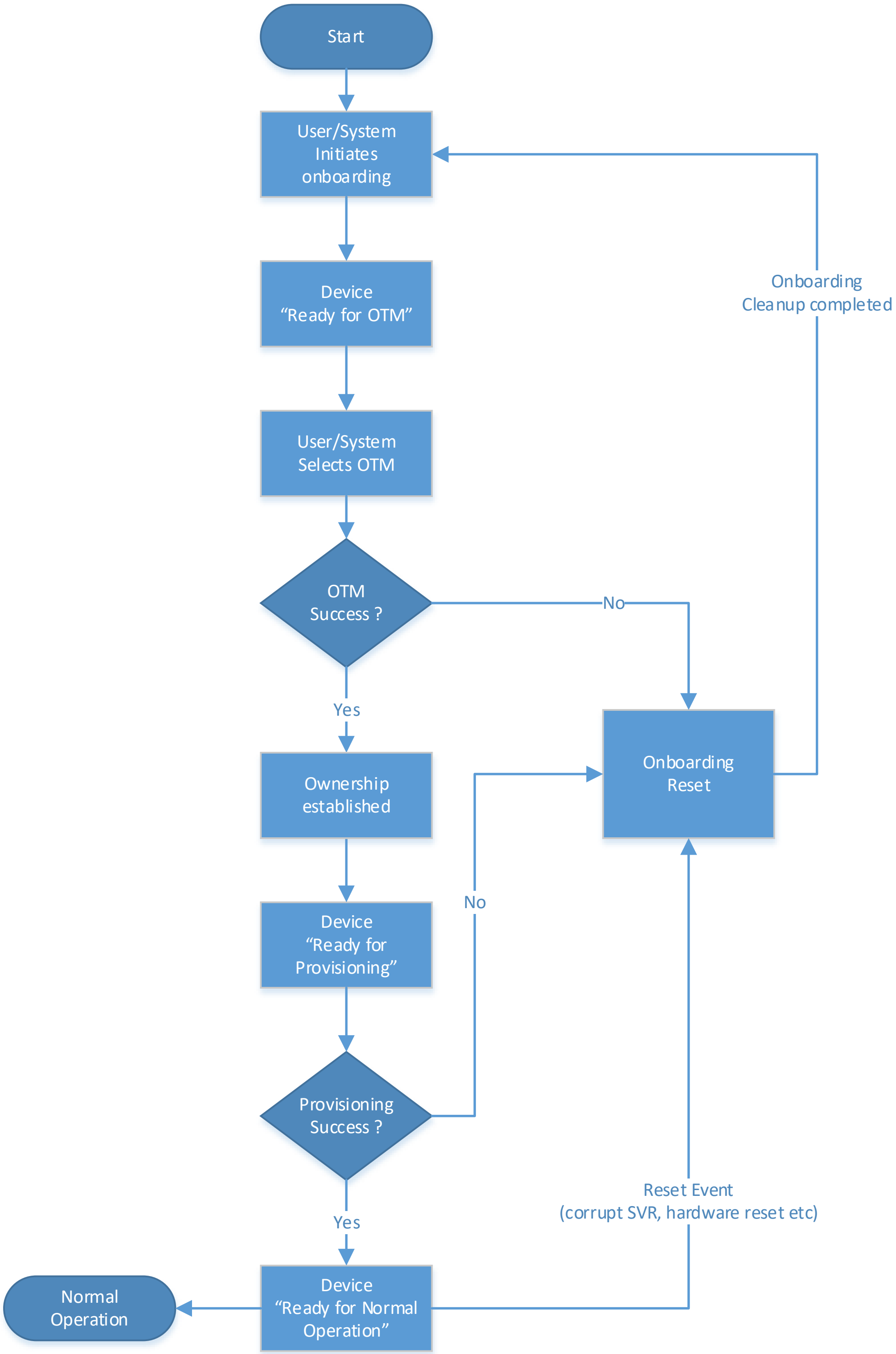

Start
User/System Initiates onboarding
Device “Ready for OTM”
User/System Selects OTM
OTM Success ?
No
Yes
Ownership established
Device “Ready for Provisioning”
Provisioning Success ?
No
Yes
Onboarding Cleanup completed
Onboarding Reset
Reset Event (corrupt SVR, hardware reset etc)
Device “Ready for Normal Operation”
Normal Operation

Figure 10 – OCF onboarding process

5.3.3 Establishing a Device Owner

The objective behind establishing Device ownership is to allow the OCF Security Domain Owner to assert itself as the owner and manager of the Device and introduce the Device into the OCF Security Domain. This is done through the use of a DOTS that includes the creation of an ownership context between the new Device and the DOTS and asserts operational control and management of the Device. The DOTS is hosted on an OBT.

The DOTS uses one of the OTMs specified in 7.3 to securely establish Device ownership.

An OTM establishes a new owner (the operator of DOTS) that is authorized to manage the Device. Ownership Transfer accomplishes the following:

– The DOTS provisions an Owner Credential (OC) to the "creds" Property in the "/oic/sec/cred" Resource of the Device. This OC allows the Device and DOTS to mutually authenticate during subsequent interactions. The OC associates the DOTS Device UUID with the "rowneruuid" Property of the "/oic/sec/doxm" Resource establishing it as the Resource owner.
– The Device owner establishes trust in the Device through the OTM.
– Provisioning of appropriate credentials for the Device to be a member of the OCF Security Domain.

5.3.4 Provisioning for Normal Operation

Once the Device has the necessary information to initiate provisioning, the next step is to provision additional security configuration that allows the Device to become operational. This may include setting various parameters and may also involve multiple steps. Also provisioning of ACL's for the various Resources hosted by the Server on the Device is done at this time. The provisioning step is not limited to this stage only. Device provisioning may happen at multiple stages in the Device's operational lifecycle. However specific security related provisioning of Resource and Property state would likely happen at this stage at the end of which, each Device reaches RFNOP. RFNOP is consistent and well defined regardless of the specific OTM used or regardless of the variability in what gets provisioned. However individual OTM mechanisms and provisioning steps may specify additional configuration of Resources and Property states. The minimal mandatory configuration required for a Device to be in RFNOP is specified in 8.

5.3.5 OCF Compliance Management System

The OCF Compliance Management System (OCMS) is a service maintained by the OCF that provides Certification status and information for OCF Devices.

The OCMS shall provide a JSON-formatted Certified Product List (CPL), hosted at the URI: https://www.openconnectivity.org/certification/ocms-cpl.json

The OBT shall possess the Root Certificate needed to enable https connection to the URI https://www.openconnectivity.org/certification/ocms-cpl.json.

The OBT should periodically refresh its copy of the CPL via the URI https://www.openconnectivity.org/certification/ocms-cpl.json, as appropriate to OCF Security Domain owner policy requirements.

5.4 Provisioning

5.4.1 Provisioning general

OCF security provisioning includes processes during and after the ownership transfer like configuration of credentials for interacting with provisioning services, configuration of any security related Resources and credentials for interacting with any services or Devices that the provisioned Device needs to contact later on.

The Device needs to engage with the CMS and AMS to be provisioned with:

– Security credentials through a CMS, which is currently assumed to be embedded in the same OBT as the DOTS.
– Access control policies and ACLs through an AMS, which is currently assumed to be embedded in the same OBT as the DOTS.

To be able to support the use of distinct device management services, some Device Secure Virtual Resources (SVRs) have an associated Resource owner identified in the Resource's rowneruuid Property.

The "rowneruuid" Property of the "/oic/sec/doxm" and "/oic/sec/pstat" Resources identifies the DOTS.

The "rowneruuid" Property of the "/oic/sec/cred" Resource identifies the CMS.

The "rowneruuid" Property of the "/oic/sec/acl2" Resource identifies the AMS.

The DOTS provisions credentials that enable secure connections between OCF Services and the new Device. The DOTS initiates client-directed provisioning by signaling the OCF Service.

5.4.2 Access control provisioning

ACL provisioning is performed over a secure connection between the AMS and its Devices. The AMS provisions the ACL by updating the Device’s ACL Resource.

5.4.3 Credential provisioning

The CMS securely provisions credentials for Device-to-Device interactions using the CMS credential provisioned by the DOTS during the onboarding procedure. The CMS is also expected to proactively monitor the credentials installed on the Device and update them when needed (e.g. close to the expiration date).

5.4.4 Role provisioning

The Servers, receiving requests for Resources they host, need to verify the role identifier(s) asserted by the Client requesting the Resource and compare that role identifier(s) with the constraints described in the Server's ACLs Thus, a Client may need to be provisioned with one or more role credentials. Once provisioned, the Client can assert the role it is using as described in 10.4.2, if it has a certificate role credential.

Each Device holds the assertable role(s) information as a Property within the Credential Resource. Each Device holds the asserted role(s) information as Properties within the Roles Resource.

All asserted roles are used in ACL enforcement. When a server has multiple roles asserted for a Client, access to a Resource is granted if it would be granted under any of the roles.

5.5 Secure Resource Manager (SRM)

SRM plays a key role in the overall security operation. In short, SRM performs both management of SVR and access control for requests to access and manipulate Resources. SRM consists of 3 main functional elements:

- A Resource manager (RM): responsible for 1) Loading SVRs from persistent storage (using PSI) as needed. 2) Supplying the Policy Engine (PE) with Resources upon request. 3) Responding to requests for SVRs. While the SVRs are in SRM memory, the SVRs are in a format that is consistent with device-specific data store format. However, the RM will use JSON format to marshal SVR data structures before being passed to PSI for storage, or travel off-device.
- A Policy Engine (PE) that takes requests for access to SVRs and based on access control policies responds to the requests with either "ACCESS_GRANTED" or "ACCESS_DENIED". To make the access decisions, the PE consults the appropriate ACL and looks for best Access Control Entry (ACE) that can serve the request given the subject (Device or role) that was authenticated by DTLS.
- Persistent Storage Interface (PSI): PSI provides a set of APIs for the RM to manipulate files in its own memory and storage. The SRM design is modular such that it may be implemented in the Platform’s secure execution environment; if available.

Figure 11 depicts OCF's SRM Architecture.

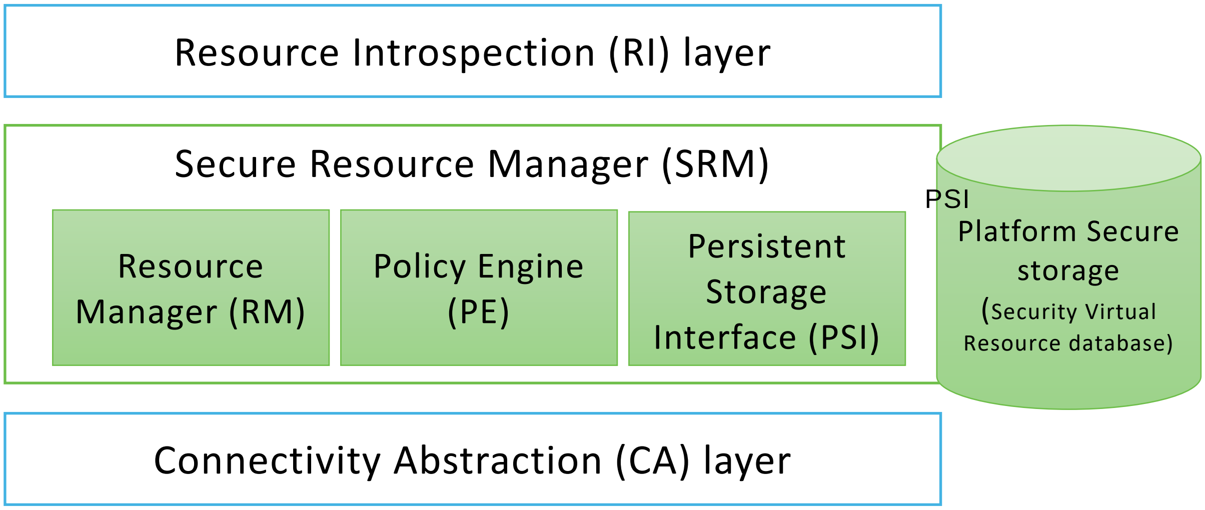

Resource Introspection (RI) layer
Secure Resource Manager (SRM)
Resource Manager (RM)
Policy Engine (PE)
Persistent Storage Interface (PSI)
PSI
Platform Secure storage
(Security Virtual Resource database)
Connectivity Abstraction (CA) layer

Figure 11 – OCF's SRM architecture

5.6 Credential overview

Devices may use credentials to prove the identity and role(s) of the parties in the Client to Server communication. Credentials may be symmetric or asymmetric. Each Device stores secret and public parts of its own credentials where applicable, as well as credentials for other Devices that have been provisioned by the DOTS or a CMS. These credentials may then be used in the establishment of secure communication sessions (e.g. using DTLS, TLS or OSCORE). Role certificates may be used after an authenticated session is established to assert one or more roles for a Device.

The credential types available within this document include:

– Pairwise symmetric keys

– Certificates

– Raw asymmetric keys

Devices may not support all of these credential types. The set of supported credential types for any Device is contained in its "sct" Property of the "/oic/sec/doxm" Resource.

5.7 Event logging

5.7.1 Event logging general

An OCF Platform can generate various kinds of Auditable Events. These Auditable Events can be used for log analysis or for real-time understanding of a system condition. Usually multiple Auditable Events are stored to backtrack problems that have occurred in the system. The storage capacity of IoT devices is typically very limited, so a specific type of data structure such as a ring buffer is often used.

An OCF Device logs Auditable Event Entries (AEE) for all Auditable Events that satisfy the "categoryfilter" and "priorityfilter" Properties of the "/oic/sec/ael" Resource. The AEEs are stored in local storage (see Figure 1). Due to the limited size of the local storage, OCF Security Domain Owner is expected to adjust the filtering options.

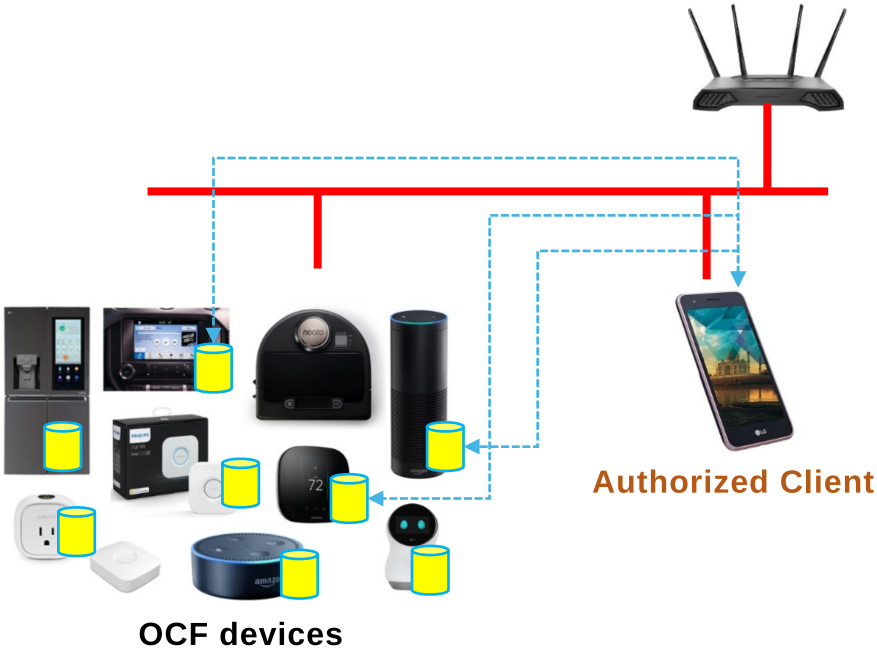


Figure 12 – Store Events in local storage

5.8 End-to-End security of unicast messages

The Security model for End-to-End Security of Unicast Messages is described in Figure 3 and Figure 4 of clause 5.1 and the accompanying steps.

OCF uses the Object Security for Constrained RESTful Environments (OSCORE) protocol IETF RFC 8613 for End-to-End Security of Unicast Messages. The Origin Client transforms a CoAP-encoded OCF CRUDN request message into an OSCORE request message which can be forwarded towards the Target Server by OCF Proxies; the Target Server then processes the OSCORE request message to extract the OCF CRUDN request message. Likewise, the Target Server then transforms a CoAP-encoded OCF CRUDN response message into an OSCORE response message which can be forwarded towards the Origin Client by OCF Proxies; the Origin Client then processes the OSCORE response message to extract the OCF CRUDN response message. OSCORE preserves the confidentiality, integrity and freshness of the OCF CRUDN messages while in transit between the Origin Client and the Target Server.

OSCORE specification supports transporting OSCORE messages using the CoAP protocol already used in OCF specifications. The payload of the OSCORE message is a CBOR Object Signature and Encryption (COSE) object (see IETF RFC 8152) in which all elements of the CoAP-encoded OCF CRUDN message, other than those parts which are needed for delivering the message to the receiving Device, are encrypted and integrity protected. OSCORE also includes replay protection.

5.9 Overview of Simple Secure Multicast

The Security model for SSM is described in Figure 6 of clause 5.1 and the accompanying steps.OCF uses the OSCORE protocol IETF RFC 8613 for the Security of SSM Messages. The Client transforms a CoAP-encoded UPDATE request message into an OSCORE request message which can be forwarded towards the Servers of the SSM Group using network-layer multicast; the Server then processes the OSCORE request message to extract the UPDATE request message.

Note: OSCORE is also used, albeit slightly differently, for End-to-End Security of Unicast Messages.

The intended use of the SSM feature is only for updating Resources with one non-confirmable multicast request. Other CRUDN operations (e.g. RETRIEVE, confirmable UPDATE, etc) are not supported because the SSM protocol is not designed to send individual responses back on the request. Hence when sending such operation by means of SSM, the individual Servers will silently ignore the request message and not send a response.

The OSCORE specification supports transporting OSCORE messages using the CoAP protocol already used in OCF specifications. The payload of the OSCORE message is a CBOR Object Signing and Encryption (COSE) object (see IETF RFC 8152) in which all elements of the CoAP-encoded UPDATE request message, other than those parts which are needed for delivering the message to the receiving Device, are encrypted and integrity protected. OSCORE also includes replay protection.

The setup of the OSCORE security context for an SSM Group is a 1-N relationship:

– the SSM Client Context of the SSM Group is only provisioned once in the Client of the SSM Group, and

– copies of the SSM Server Context of the SSM Group are provisioned to one or more Servers in the SSM Group.

Figure 13 depicts the relationship of the SSM Client Context and SSM Server Context.

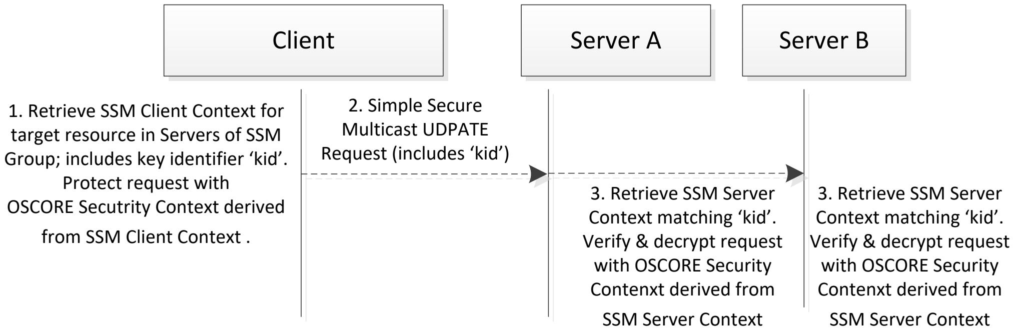

Client
Server A
Server B
1. Retrieve SSM Client Context for target resource in Servers of SSM Group; includes key identifier ‘kid’. Protect request with OSCORE Secutrity Context derived from SSM Client Context .
2. Simple Secure Multicast UDPATE Request (includes ‘kid’)
3. Retrieve SSM Server Context matching ‘kid’. Verify & decrypt request with OSCORE Security Contenxt derived from SSM Server Context
3. Retrieve SSM Server Context matching ‘kid’. Verify & decrypt request with OSCORE Security Contenxt derived from SSM Server Context

Figure 13 – Relationship diagram for Simple Secure Multicast messages

Figure 14 depicts the full setup and usage.

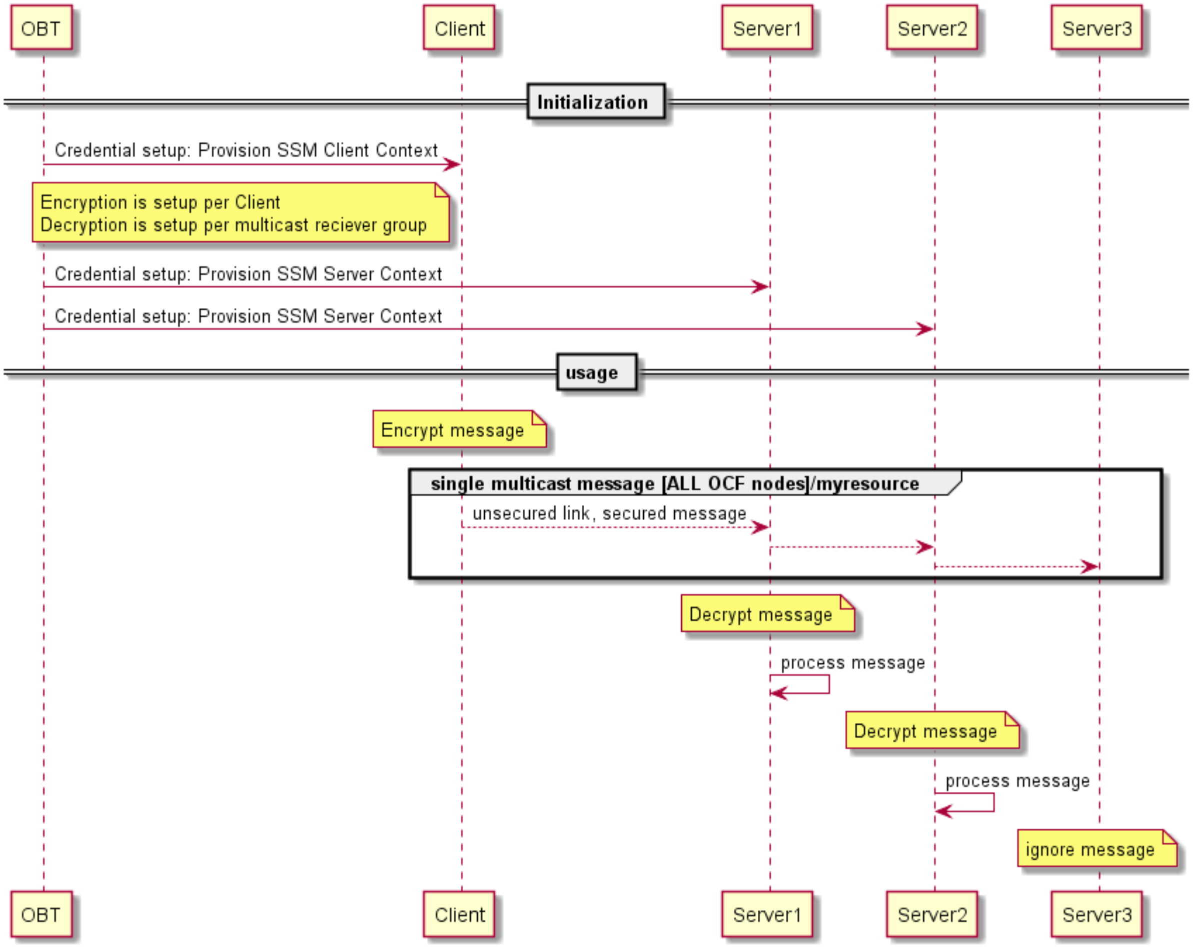

OBT
Client
Server1
Server2
Server3
Initialization
Credential setup: Provision SSM Client Context
Encryption is setup per Client
Decryption is setup per multicast reciever group
Credential setup: Provision SSM Server Context
Credential setup: Provision SSM Server Context
usage
Encrypt message
single multicast message [ALL OCF nodes]/myresource
unsecured link, secured message
Decrypt message
process message
Decrypt message
process message
ignore message
OBT
Client
Server1
Server2
Server3

Figure 14 – Setup and usage of Secure Simple Multicast

The first message after onboarding is implicitly trusted by the Server as being a valid message. This is due to the replay window not yet being set up by the Server. The Server stores the received information so that the replay protection is enabled after receiving the first message.

6 Security for the Discovery process

6.1 Preamble

The main function of a discovery mechanism is to provide Universal Resource Identifiers (URIs, called links) for the Resources hosted by the Server, complemented by attributes about those Resources and possible further link relations. (in accordance to clause 10 in ISO/IEC 30118-1)

6.2 Security considerations for Discovery

When defining discovery process, care must be taken that only a minimum set of Resources are exposed to the discovering entity without violating security of sensitive information or privacy requirements of the application at hand. This includes both data included in the Resources, as well as the corresponding metadata.

To achieve extensibility and scalability, this document does not provide a mandate on discoverability of each individual Resource. Instead, the Server holding the Resource will rely on ACLs for each Resource to determine if the requester (the Client) is authorized to see/handle any of the Resources.

The "/oic/sec/acl2" Resource contains ACL entries governing access to the Server hosted Resources. (See 13.5)

Aside from the privacy and discoverability of Resources from ACL point of view, the discovery process itself needs to be secured. This document sets the following requirements for the discovery process:

1) Providing integrity protection for discovered Resources.
2) Providing confidentiality protection for discovered Resources that are considered sensitive.

The discovery of Resources is done by doing a RETRIEVE operation (either unicast or multicast) on the known "/oic/res" Resource.

The discovery request is sent over a non-secure channel (multicast or unicast without DTLS), a Server cannot determine the identity of the requester. In such cases, a Server that wants to authenticate the Client before responding can list the secure discovery URI (e.g. coaps://IP:PORT/oic/res) in the unsecured "/oic/res" Resource response. This means the secure discovery URI is by default discoverable by any Client. The Client will then be required to send a separate unicast request using DTLS to the secure discovery URI.

For example, a Client with Device UUID "d1" (UUID:"0685B960-736F-46F7-BEC0-9E6CBD61ADC1") makes a RETRIEVE request on the "/door" Resource hosted on a Server with Device UUID "d3" where d3 has the ACL2s:

```
{
  "aclist2": [
    {
      "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},
      "resources": [{"href":"/door"}],
      "permission": 2, // RETRIEVE
      "aceid": 1
    },
    {
      "subject": {"authority": "owner", "role": "owner"}
      "resources": [{"href":"/door"}],
      "permission": 2, // RETRIEVE
      "aceid": 2
    },
    {
      "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},
      "resources": [{"href":"/door/lock"}],
      "permission": 4, // UPDATE
      "aceid": 3
    }
  ],
  "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
}
```

The ACL indicates that Client "d1" has RETRIEVE permissions on the Resource. Hence when device "d1" does a discovery on the "/door" Resource of the Server "d3", the response will include all the URIs in the "/door" Resource. Client "d2" without a Role ID "owner" will get an error response that includes no URI.

Discovery results delivered to d1 regarding d3’s "/door" Resource from the secure interface:

```
[
  {
    "href": "/door",
    “rel”: “self”,
    "rt": ["oic.wk.col"],
    "if": ["oic.if.ll", “oic.if.b”, "oic.if.baseline"],
     "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  },
  {
    "href": "/door/lock",
    "rt": ["oic.r.lock.status "],
    "if": ["oic.if.a", “oic.if.baseline”],
    "eps":[{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
  }
]
```

7 Security provisioning

7.1 Device identity

7.1.1 General Device identity

A Device shall be identified by a Device UUID value that is established as part of the device onboarding and contained in the "deviceuuid" Property of the "/oic/sec/doxm" Resource. Device UUIDs shall be unique within the scope of the corresponding OCF Security Domain, and are expected to be randomly generated and provisioned by the OBT. The DOTS is expected to verify that the chosen new Device UUID does not conflict with Device UUIDs previously introduced into the OCF Security Domain.

Devices maintain an association of their Device UUIDs and their own cryptographic credential(s) via "/oic/sec/cred" Resource. The identity is cryptographically bound in case of a certificate credential, or is bound via internal mappings in the "/oic/sec/cred" Resource otherwise. The "/oic/sec/cred" Resource maintains a list of a Device's own and other Device's credentials. Multiple credentials may be associated with the same Device UUID.A Device is expected to only present credentials associated with its own Device UUID for peer authentication purposes. Devices regard the "/oic/sec/cred" Resource as authoritative when verifying authentication credentials of a peer Device.

In case of an authenticated connection, the Device UUID is treated as a Client's identity for purposes of the Access Control check for the target Resource. The Device UUID of a Client is matched against the Subject UUIDs in the pre-provisioned entries of Server's "/oic/sec/acl2" Resource. The Server determines Client's Device UUID based on the credential used for the establishment of the session.

An OCF Platform, which may host multiple Devices, is identified by a Platform ID. The Platform ID is globally unique and inserted in the device in an integrity protected manner (e.g. inside secure storage or signed and verified).

An OCF Platform may have a secure execution environment, used to secure unique identifiers and secrets. If a Platform hosts multiple Devices, some mechanism is needed to provide each Device with the appropriate and separate security context.

7.1.2 Device identity for Devices with UAID [Deprecated]

This clause is intentionally left blank.

7.2 Device ownership

This is an informative clause. Devices are logical entities that are security endpoints that have an identity that is authenticable using cryptographic credentials. A Device is Unowned when it is first initialized. Establishing device ownership is a process by which the device asserts its identity to the DOTS and the DOTS provisions an owner identity. This exchange results in the device changing its ownership state, thereby preventing a different DOTS from asserting administrative control over the device.

The ownership transfer process starts with the OBT discovering a new device that is in Unowned state through examination of the "Owned" Property of the "/oic/sec/doxm" Resource of the new device. At the end of ownership transfer, the following is accomplished:

1) The DOTS establishes a secure session with new device.
2) Optionally asserts any of the following:
    a) Proximity (using PIN) of the OBT to the Platform.
    b) Manufacturer's certificate asserting Platform vendor, model and other Platform specific attributes.

3) Determines the device identifier.
4) Determines the device owner.
5) Specifies the device owner (e.g. Device UUID of the OBT).
6) Provisions the device with owner’s credentials.
7) Sets the "Owned" state of the new device to TRUE.

.

7.3 Device Ownership Transfer Methods

7.3.1 OTM implementation requirements

This document provides specifications for several methods for ownership transfer. Implementation of each individual ownership transfer method is considered optional. However, each device shall implement at least one of the ownership transfer methods not including vendor specific methods.

All OTMs included in this document are considered optional. Each vendor is required to choose and implement at least one of the OTMs specified in this document. The OCF, does however, anticipate vendor-specific approaches will exist. Should the vendor wish to have interoperability between a vendor-specific OTM and OBTs from other vendors, the vendor must work directly with OBT vendors to ensure interoperability. Notwithstanding, standardization of OTMs is the preferred approach. In such cases, a set of guidelines is provided in 7.3.7 to help vendors in designing vendor-specific OTMs.

The "/oic/sec/doxm" Resource is extensible to accommodate vendor-defined owner transfer methods (OTM). The DOTS determines which OTM is most appropriate to onboard the new Device. All OTMs shall represent the onboarding capabilities of the Device using the "oxms" Property of the "/oic/sec/doxm" Resource. The DOTS determines the Device’s supported credential types using the Supported Credential Types "sct" Property of the "/oic/sec/doxm" Resource. The DOTS and CMS provision credentials according to the credential types supported.

Figure 15 depicts new Device discovery sequence.

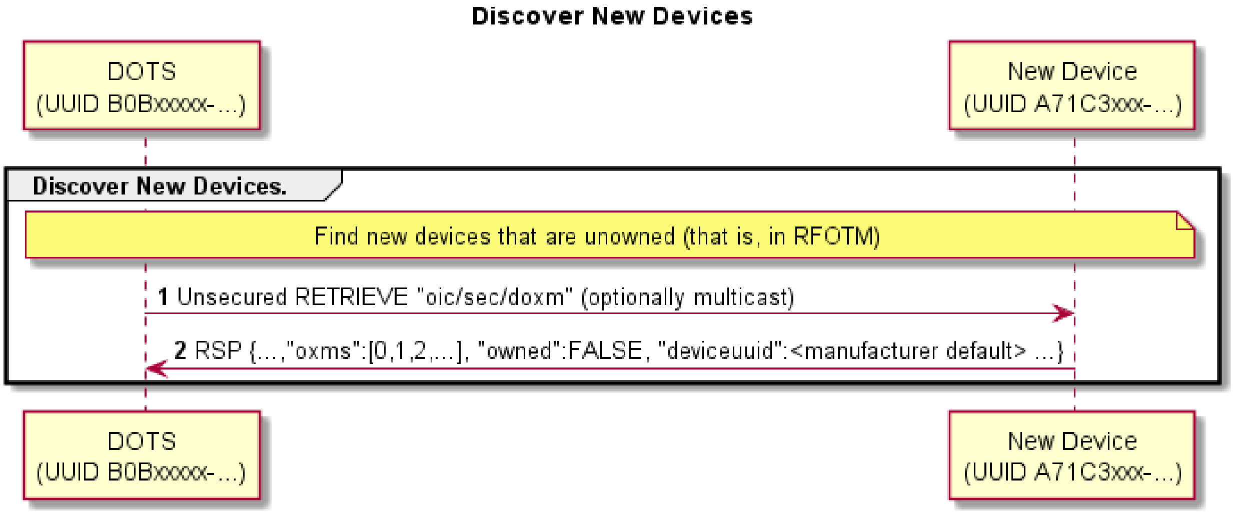

Discover New Devices
DOTS
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)
Discover New Devices.
Find new devices that are unowned (that is, in RFOTM)
1 Unsecured RETRIEVE "oic/sec/doxm" (optionally multicast)
2 RSP {...,"oxms":[0,1,2,...], "owned":FALSE, "deviceuuid":<manufacturer default> ...}
DOTS
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)

Figure 15 – Discover new Device sequence

Table 1 – Discover new Device details

Step	Description
1	The DOTS queries to see if the new device is not yet owned.
2	The new device returns the "/oic/sec/doxm" Resource containing ownership status and supported OTMs. It also contains a temporal Device UUID that may change subsequent to successful owner transfer. The device should supply a temporal ID to facilitate discovery as a guest device. Refer to OCF Onboarding Tool Specification for security considerations regarding selecting an OTM.

A Device shall support selective use of unsecured multicast to receive RETRIEVE requests to the Device "/oic/sec/doxm" Resource, as shown in Figure 15. Clause 10.4 of the ISO/IEC 30118-1 provides the generic details for using CoAP multicast requests in OCF. Multicast retrieval of the "/oic/sec/doxm" Resource supports filtering using the "owned" query parameter. When a multicast RETRIEVE request omits the "owned" query parameter or includes the "owned" query parameter set to “false”, then the Device shall respond only if the Device is in RFOTM and there is no open Device Onboarding Connection. Otherwise the request shall be ignored by the Device, regardless of ACE configuration.

Vendor-specific device OTMs shall adhere to the "/oic/sec/doxm" Resource Specification for OCs that results from vendor-specific device OTM. Vendor-specific OTM should include provisions for establishing trust in the new Device by the DOTS and optionally establishing trust in the OBT by the new Device.

The new device may have to perform some initialization steps at the beginning of an OTM. For example, if the Random PIN Based OTM is initiated, the new device may generate a random PIN value. The DOTS updates the oxmsel property of "/oic/sec/doxm" to the value corresponding to the OTM being used, before performing other OTM steps. This update notifies the new device that ownership transfer is starting.

The end state of a vendor-specific OTM shall allow the new Device to authenticate to the OBT and the OBT to authenticate to the new device.

Additional provisioning steps may be performed subsequent to owner transfer success leveraging the established OTM session.

7.3.2 SharedKey credential calculation

The SharedKey credential is derived using a PRF that accepts the key_block value resulting from the DTLS handshake used for onboarding. The new Device shall use the following calculation to ensure interoperability across vendor products (the DOTS performs the same calculation):

SharedKey = *PRF*(Secret, Message);

Where:

- PRF shall use TLS 1.2 PRF defined by IETF RFC 5246 clause 5.
- Secret is the key_block resulting from the DTLS handshake
  - See IETF RFC 5246 clause 6.3
  - The length of key_block depends on cipher suite.
    - (e.g. 96 bytes for TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 40 bytes for TLS_PSK_WITH_AES_128_CCM_8)
- Message is a concatenation of the following:
  - DoxmType string for the current onboarding method (e.g. "oic.sec.doxm.jw")
    - See clause 13.2.2 for specific DoxmTypes

    - ▪ Owner ID is a UUID identifying the device owner identifier and the device that maintains SharedKey.
        - Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
    - ▪ Device UUID is new device's UUID
        - Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
- SharedKey Length will be 32 octets.
    - ▪ If subsequent DTLS sessions use 128 bit encryption cipher suites the left most 16 octets will be used. DTLS sessions using 256-bit encryption cipher suites will use all 32 octets.

7.3.3 Certificate credential generation

The Certificate Credential will be used by Devices for secure bidirectional communication. The certificates will be issued by a CMS or an external certificate authority (CA). This CA will be used to mutually establish the authenticity of the Device.

7.3.4 Just-Works OTM

7.3.4.1 Just-Works OTM general

Just-works OTM creates a symmetric key credential that is a pre-shared key used to establish a secure connection through which a device should be provisioned for use within the owner's OCF Security Domain. Provisioning additional credentials and Resources is a typical step following ownership establishment. The pre-shared key is called SharedKey.

The DOTS selects the Just-works OTM using the "oxmsel" Property of the "/oic/sec/doxm" Resource and establishes a DTLS session using a cipher suite defined for the Just-works OTM.

Just Works OTM sequence is shown in Figure 16 and steps described in Table 2.

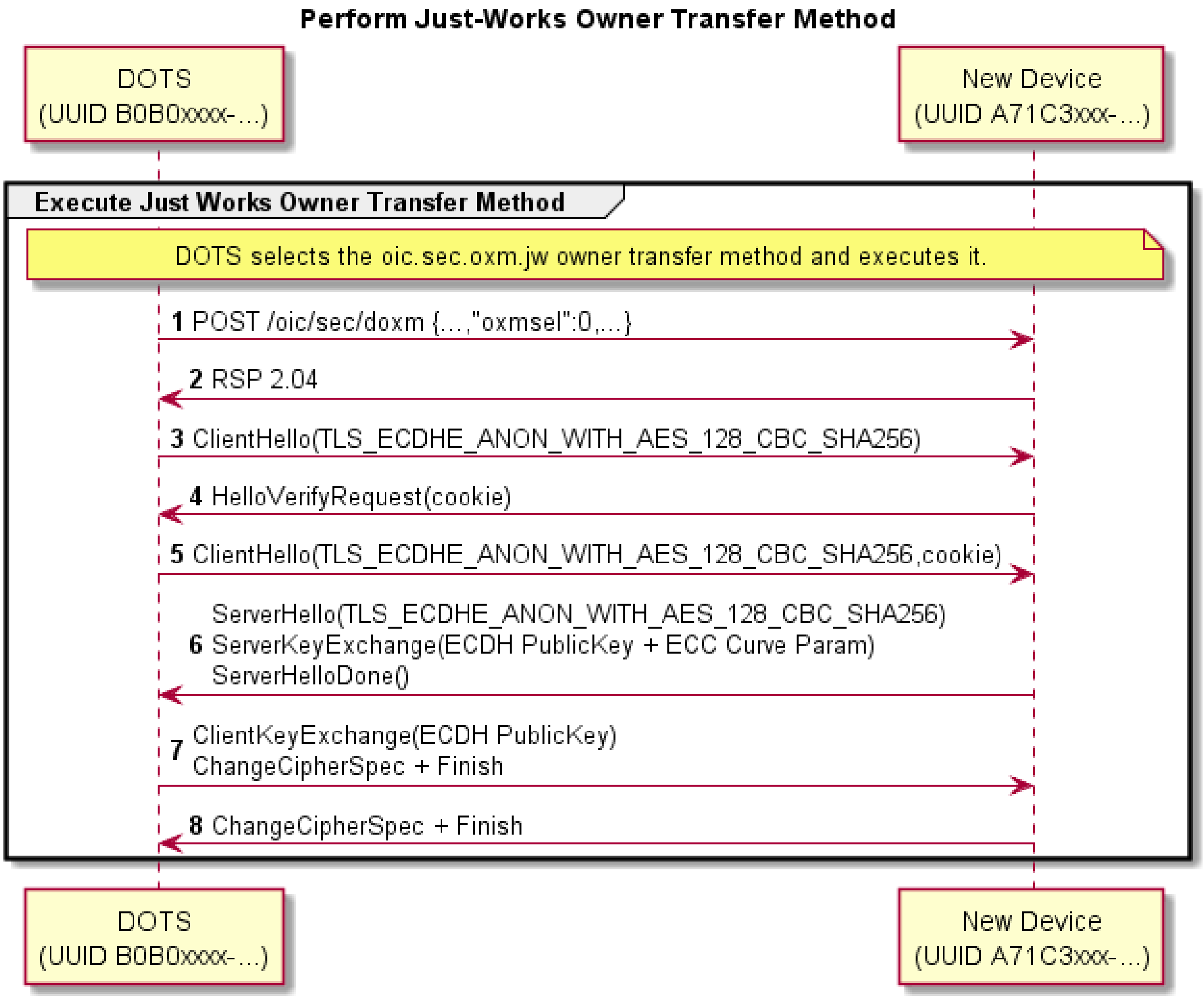

Perform Just-Works Owner Transfer Method
DOTS
(UUID B0B0xxxx-...)
New Device
(UUID A71C3xxx-...)
Execute Just Works Owner Transfer Method
DOTS selects the oic.sec.oxm.jw owner transfer method and executes it.
1 POST /oic/sec/doxm {...,"oxmsel":0,...}
2 RSP 2.04
3 ClientHello(TLS_ECDHE_ANON_WITH_AES_128_CBC_SHA256)
4 HelloVerifyRequest(cookie)
5 ClientHello(TLS_ECDHE_ANON_WITH_AES_128_CBC_SHA256,cookie)
ServerHello(TLS_ECDHE_ANON_WITH_AES_128_CBC_SHA256)
6 ServerKeyExchange(ECDH PublicKey + ECC Curve Param)
ServerHelloDone()
7 ClientKeyExchange(ECDH PublicKey)
ChangeCipherSpec + Finish
8 ChangeCipherSpec + Finish
DOTS
(UUID B0B0xxxx-...)
New Device
(UUID A71C3xxx-...)

Figure 16 – A Just Works OTM

Table 2 – A Just Works OTM details

Step	Description
1, 2	The DOTS notifies the Device that it selected the "Just Works" method.
3 - 8	A DTLS session is established using anonymous Diffie-Hellman.[a]
[a] This method assumes the operator is aware of the potential for man-in-the-middle attack and has taken precautions to perform the method in a clean-room network.	

7.3.4.2 Security considerations

Anonymous Diffie-Hellman key agreement is subject to a man-in-the-middle attacker. Use of this method presumes that both the DOTS and the new device perform the "just-works" method assumes onboarding happens in a relatively safe environment absent of an attack device.

This method doesn’t have a trustworthy way to prove the Device UUID asserted is reliably bound to the device.

The new device should use a temporal Device UUID prior to transitioning to an owned device while it is considered a guest device to prevent privacy sensitive tracking. The device asserts a non-temporal Device UUID that could differ from the temporal value during the secure session in which owner transfer exchange takes place. The DOTS verifies the asserted Device UUID does not conflict with a Device UUID already in use. If it is already in use the existing credentials are used to establish a secure session.

An un-owned Device that also has established device credentials might be an indication of a corrupted or compromised device.

7.3.5 Random PIN based OTM

7.3.5.1 Random PIN based OTM general

The Random PIN method establishes physical proximity between the new device and the OBT can prevent man-in-the-middle attacks. The Device generates a random number that is communicated to the DOTS over an Out of Band Communication Channel. The definition of an Out of Band Communication Channel is outside the scope of the definition of device OTMs. The DOTS and new Device use the PIN in a key exchange as evidence that an End User authorized the transfer of ownership by having physical access to the new Device via the Out-of-Band Communication Channel.

7.3.5.2 Random PIN based Owner Transfer sequence

Random PIN-based OTM sequence is shown in Figure 17 and steps described in Table 3.

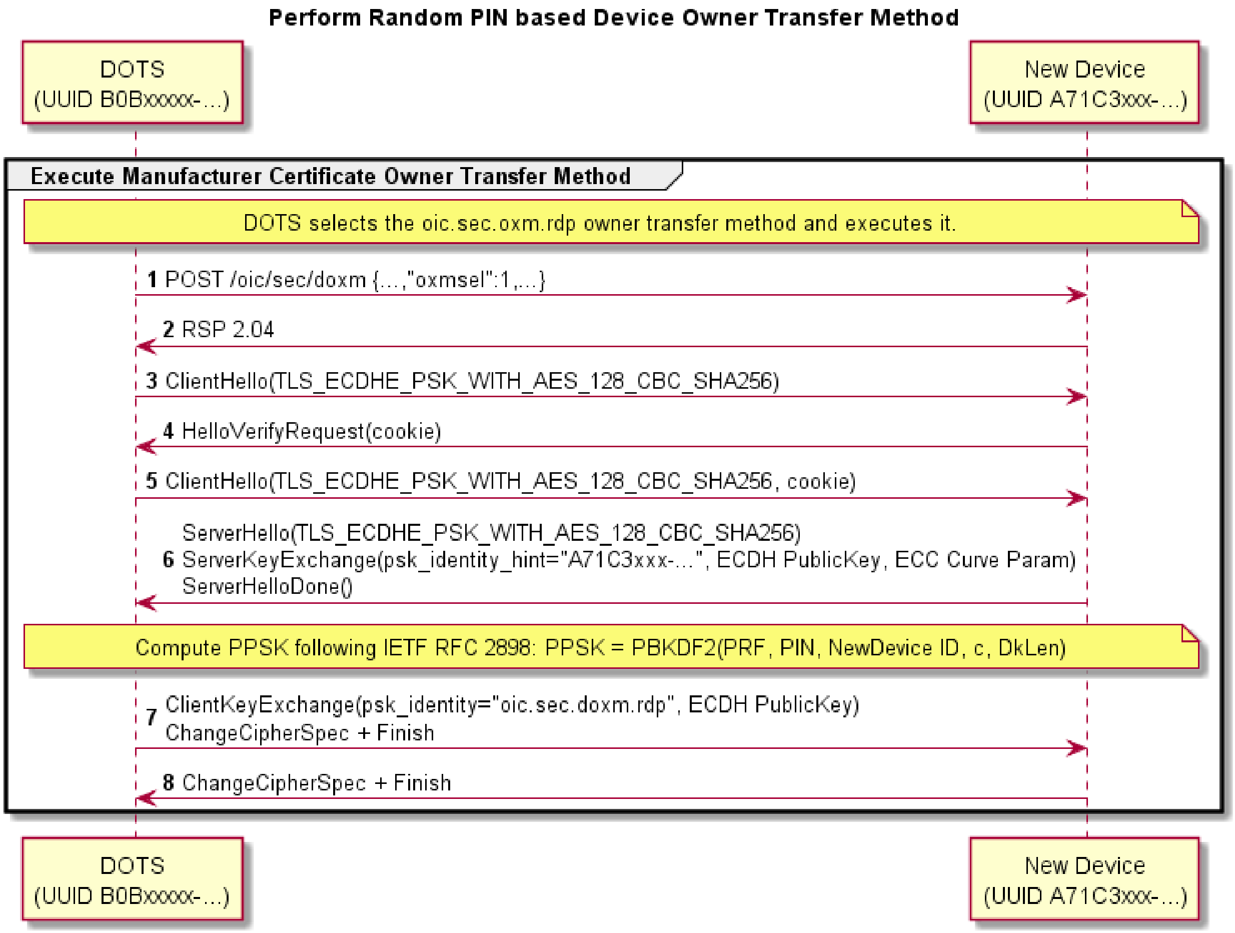

Perform Random PIN based Device Owner Transfer Method
DOTS
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)
Execute Manufacturer Certificate Owner Transfer Method
DOTS selects the oic.sec.oxm.rdp owner transfer method and executes it.
1 POST /oic/sec/doxm {...,"oxmsel":1,...}
2 RSP 2.04
3 ClientHello(TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256)
4 HelloVerifyRequest(cookie)
5 ClientHello(TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256, cookie)
ServerHello(TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256)
6 ServerKeyExchange(psk_identity_hint="A71C3xxx-...", ECDH PublicKey, ECC Curve Param)
ServerHelloDone()
Compute PPSK following IETF RFC 2898: PPSK = PBKDF2(PRF, PIN, NewDevice ID, c, DkLen)
7 ClientKeyExchange(psk_identity="oic.sec.doxm.rdp", ECDH PublicKey)
ChangeCipherSpec + Finish
8 ChangeCipherSpec + Finish
DOTS
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)

Figure 17 – Random PIN-based OTM

Table 3 – Random PIN-based OTM details

Step	Description
1, 2	The DOTS notifies the Device that it selected the "Random PIN" method.
3 - 8	A DTLS session is established using PSK-based Diffie-Hellman cipher suite. The PIN is supplied as the PSK parameter. The PIN is randomly generated by the new device then communicated via an Out of Band Communication Channel that establishes proximal context between the new device and the DOTS. The security principle is the attack device will be unable to intercept the PIN due to a lack of proximity.

The following requirements apply to the DTLS handshake messages for this OTM:

– At step 6:

  – The Server shall only use a DTLS ciphersuite supported by the Random PIN Based OTM (see clause 11.3.2.2),

  - The new Device shall set the "psk_identity_hint" field of the ServerKeyExchange message to the concatenation of
    - the string "oic.sec.doxm.rdp";
    - the colon character ':';
  - The "deviceuuid" Property of the "/oic/sec/doxm" Resource being sent in responses when the new Device is in RFOTM and when a Device Onboarding Connection is not currently established.
- At step 7:
  - If the new Device determines that the "psk_identity" field of the ClientKeyExchange message does not matche the string "oic.sec.doxm.rdp", then the new Device shall reject the DTLS Handshake.
  - the new Device shall apply the key derivation below.

NOTE The string "oic.sec.doxm.rdp" is the URN defined for the Random PIN-based OTM in Table 19 and is included to allow future OTMs to re-use the DTLS cipher suites without confusion about which OTM should be applied.

This OTM uses a pseudo-random function (PBKDF2) defined by IETF RFC 2898 and a PIN exchanged via an Out of Band Communication Channelto generate a pre-shared key. The PIN-authenticated pre-shared key (PPSK) is supplied to TLS cipher suites that accept a PSK.

- PPSK = PBKDF2(PRF, PIN, Device UUID, c, dkLen)

The PBKDF2 function has the following parameters:

- PRF – Uses the TLS 1.2 PRF defined by IETF RFC 5246.
- PIN – obtained via Out of Band Communication Channel.
- Device UUID – the "deviceuuid" Property of the "/oic/sec/doxm" Resource being sent in responses when the new Device is in RFOTM and when a Device Onboarding Connection is not currently established.

Use raw bytes as specified in IETF RFC 4122 clause 4.1.2

- c – Iteration count initialized to 1000
- dkLen – Desired length of the derived PSK in octets.

7.3.5.3 Security considerations

Security of the Random PIN mechanism depends on the entropy of the PIN. Using a PIN with insufficient entropy may allow a man-in-the-middle attack to recover any long-term credentials provisioned as a part of onboarding. In particular, learning the provisioned symmetric key credentials allows an attacker to masquerade as the onboarded device.

It is recommended that the entropy of the PIN be enough to withstand an online brute-force attack, 40 bits or more. For example, a 12-digit numeric PIN, or an 8-character alphanumeric (0-9a-z), or a 7-character case-sensitive alphanumeric PIN (0-9a-zA-Z). A man-in-the-middle attack is when the attacker is active on the network and can intercept and modify messages between the DOTS and device. In the man-in-the-middle attack, the attacker must recover the PIN from the key exchange messages in "real time", i.e., before the peer's time out and abort the connection attempt. Having recovered the PIN, he can complete the authentication step of key exchange. The guidance given here calls for a minimum of 40 bits of entropy, however, the assurance this provides depends on the resources available to the attacker. Given the parallelizable nature of a brute force guessing attack, the attack enjoys a linear speedup as more cores/threads are added. A more conservative amount of entropy would be 64 bits. Since the Random PIN OTM requires using a DTLS cipher suite that includes an ECDHE key exchange, the security of the Random PIN OTM is always at least equivalent to the security of the JustWorks OTM.

The Random PIN OTM also has an option to use PBKDF2 to derive key material from the PIN. The rationale is to increase the cost of a brute force attack, by increasing the cost of each guess in the attack by a tuneable amount (the number of PBKDF2 iterations). In theory, this is an effective way to reduce the entropy requirement of the PIN. Unfortunately, it is difficult to quantify the reduction, since an X-fold increase in time spent by the honest peers does not directly translate to an X-fold increase in time by the attacker. This asymmetry is because the attacker may use specialized implementations and hardware not available to honest peers. For this reason, when deciding how much entropy to use for a PIN, it is recommended that implementers assume PBKDF2 provides no security, and ensure the PIN has sufficient entropy.

The Random PIN device OTM security depends on an assumption that a secure Out of Band Communication Channel for communicating a randomly generated PIN from the new device to the OBT exists. If the Out of Band Communication Channel leaks some or the entire PIN to an attacker, this reduces the entropy of the PIN, and the attacks described above apply. The Out of Band Communication Channel should be chosen such that it requires proximity between the DOTS and the new device. The attacker is assumed to not have compromised the Out of Band Communication Channel. As an example Out of Band Communication Channel, the device may display a PIN to be entered into the OBT software. Another example is for the device to encode the PIN as a 2D barcode and display it for a camera on the DOTS device to capture and decode.

7.3.6 Manufacturer Certificate Based OTM

7.3.6.1 Manufacturer Certificate Based OTM general

The manufacturer certificate-based OTM shall use a certificate embedded into the device by the manufacturer and may use a signed OBT, which determines the Trust Anchor between the device and the DOTS.

Manufacturer embedded certificates do not necessarily need to chain to an OCF Root CA trust anchor.

For some environments, policies or administrators, additional information about device characteristics may be sought. This list of additional attestations that OCF may or may not have tested (understanding that some attestations are incapable of testing or for which testing may be infeasible or economically unviable) can be found under the OCF Security Claims x509.v3 extension described in 9.4.2.2.6.

When utilizing certificate-based ownership transfer, devices shall utilize asymmetric keys with certificate data to authenticate their identities with the DOTS in the process of bringing a new device into operation on an OCF Security Domain. The onboarding process involves several discrete steps:

1) Pre-on-board conditions
   a) The credential element of the Device’s credential Resource ("/oic/sec/cred") containing the manufacturer certificate shall be identified by the "credusage" Property containing the string "oic.sec.cred.mfgcert" to indicate that the credential contains a manufacturer certificate.
   b) The manufacturer certificate chain shall be contained in the identified credential element’s "publicdata" Property.
   c) The device shall contain a unique and immutable ECC asymmetric key pair.
   d) If the device requires authentication of the DOTS as part of ownership transfer, it is presumed that the DOTS has been registered and has obtained a certificate for its unique and immutable ECC asymmetric key pair signed by the predetermined Trust Anchor.
   e) An End User has configured the DOTS app with network access info and account info (if any).
2) The DOTS authenticates the Device using ECDSA to verify the signature. Additionally, the Device may authenticate the DOTS to verify the DOTS signature.

3) If authentication fails, the Device shall indicate the reason for failure and return to the RFOTM. If authentication succeeds, the Device shall establish an encrypted link with the DOTS in accordance with the negotiated cipher suite.

7.3.6.2 Certificate Profiles

See 9.4.2 for details.

7.3.6.3 Certificate Owner Transfer sequence security considerations

The OBT shall authenticate the device during onboarding. The device will not authenticate the OBT. During the DTLS handshake the server shall not send a Certificate Request.

7.3.6.4 Manufacturer Certificate based OTM sequence

Manufacturer Certificate Based OTM sequence is shown in Figure 18 and steps described in Table 4.

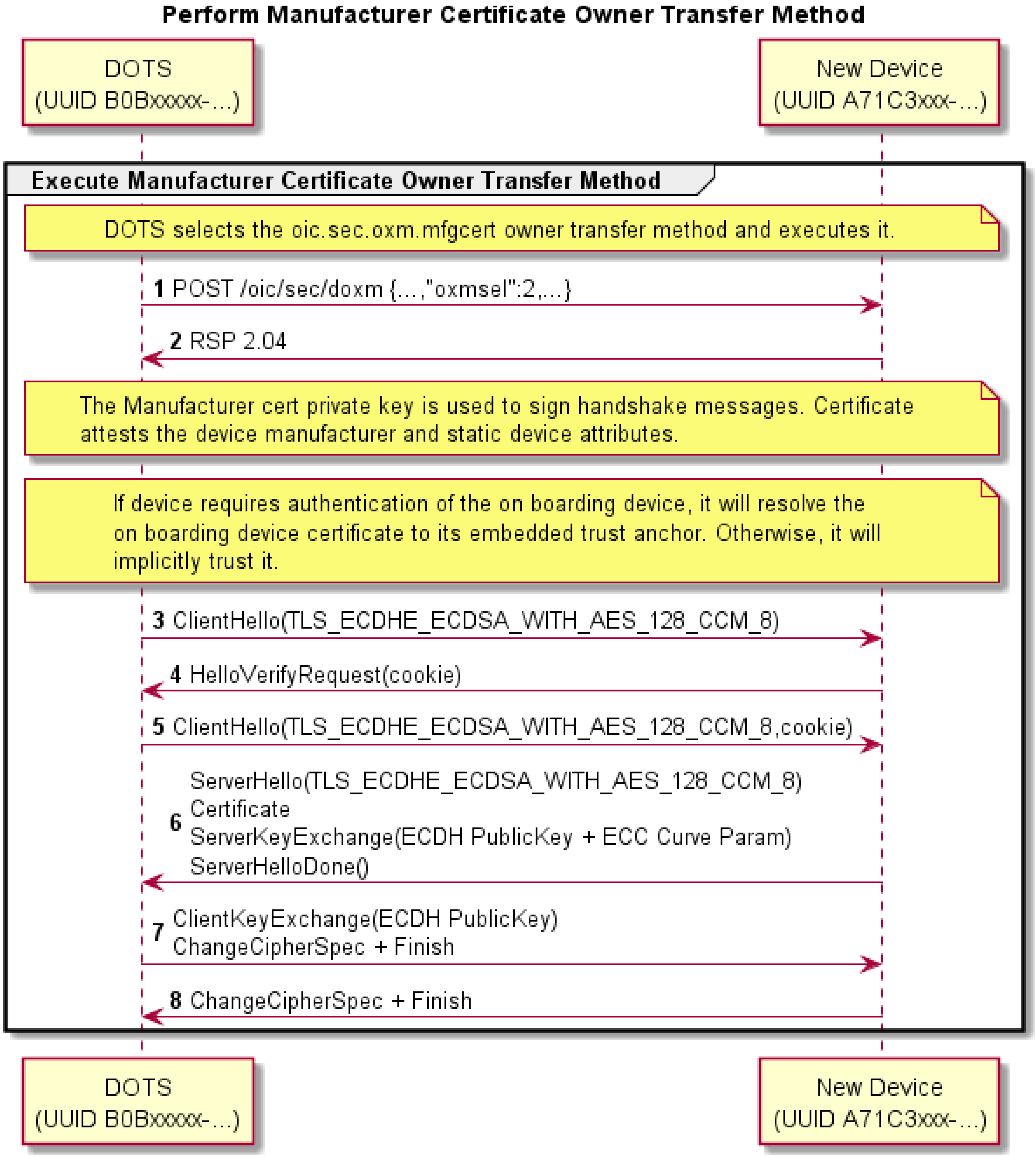

Perform Manufacturer Certificate Owner Transfer Method
DOTS
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)
Execute Manufacturer Certificate Owner Transfer Method
DOTS selects the oic.sec.oxm.mfgcert owner transfer method and executes it.
1 POST /oic/sec/doxm {...,"oxmsel":2,...}
2 RSP 2.04
The Manufacturer cert private key is used to sign handshake messages. Certificate attests the device manufacturer and static device attributes.
If device requires authentication of the on boarding device, it will resolve the on boarding device certificate to its embedded trust anchor. Otherwise, it will implicitly trust it.
3 ClientHello(TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8)
4 HelloVerifyRequest(cookie)
5 ClientHello(TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,cookie)
6 ServerHello(TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8)
Certificate
ServerKeyExchange(ECDH PublicKey + ECC Curve Param)
ServerHelloDone()
7 ClientKeyExchange(ECDH PublicKey)
ChangeCipherSpec + Finish
8 ChangeCipherSpec + Finish
DOTS
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)

Figure 18 – Manufacturer Certificate Based OTM Sequence

Table 4 – Manufacturer Certificate Based OTM Details

Step	Description
1, 2	The DOTS notifies the Device that it selected the "Manufacturer Certificate" method.

3 - 8	A DTLS session is established using the device’s manufacturer certificate. The device’s manufacturer certificate may contain data attesting to the Device hardening and security properties.

If the Manufacturer Certificate Based OTM is selected at step 1, then the following requirements apply:

- At step 6:
  - The new Device shall use a DTLS ciphersuite supported for use with the Manufacturer Certificate Based OTM (see clause 11.3.2.3),
  - The new Device shall not send a CertificateRequest message.

NOTE: CertificateRequest message is sent when establishing the DTLS connection for Device authentication using certificates (clause 10.4.1).

7.3.6.5 Security considerations

The manufacturer certificate private key is embedded in the Platform with a sufficient degree of assurance that the private key cannot be compromised.

The Platform manufacturer issues the manufacturer certificate and attests the private key protection mechanism.

7.3.7 Vendor specific OTMs

7.3.7.1 Vendor specific OTM general

The OCF anticipates situations where a vendor will need to implement an OTM that accommodates manufacturing or Device constraints. The Device OTM Resource is extensible for this purpose. Vendor-specific OTMs shall adhere to a set of conventions that all OTMs follow.

- The OBT may determine which credential types are supported by the Device. This is accomplished by querying the Device’s "/oic/sec/doxm" Resource to identify supported credential types.
- The OBT provisions the Device with OC(s).
- The OBT supplies the Device UUID and credentials for subsequent access to the OBT.
- The OBT may perform additional provisioning steps.

7.3.7.2 Vendor-specific Owner Transfer Sequence Example

Vendor-specific OTM sequence example is shown in Figure 19 and steps described in Table 5.

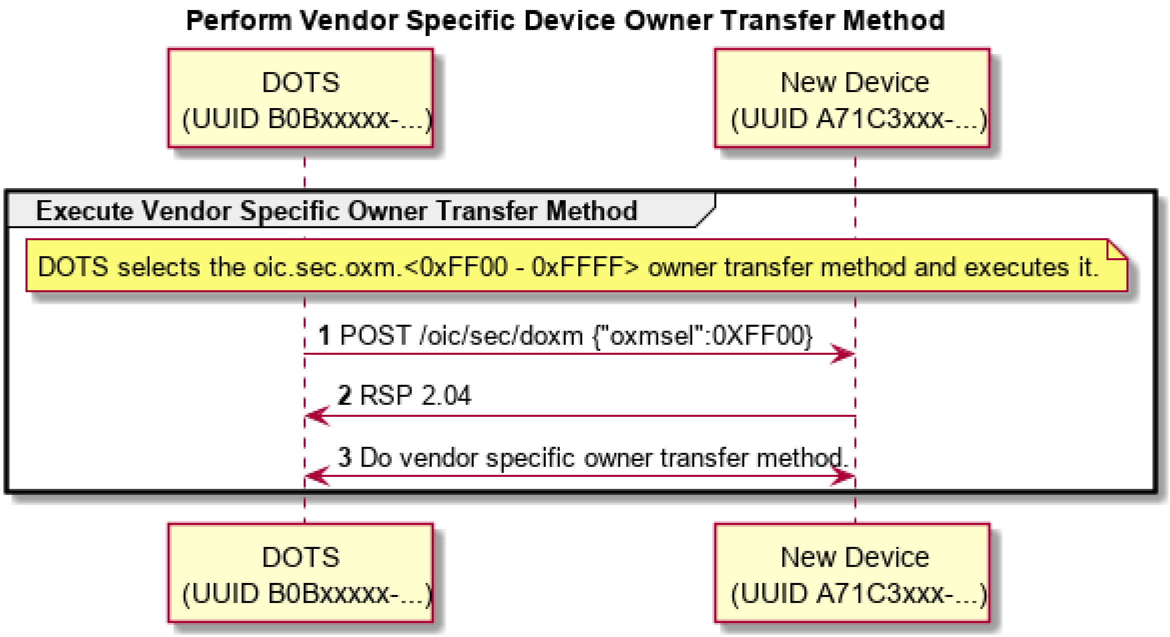

Perform Vendor Specific Device Owner Transfer Method
DOTS
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)
Execute Vendor Specific Owner Transfer Method
DOTS selects the oic.sec.oxm.<0xFF00 - 0xFFFF> owner transfer method and executes it.
1 POST /oic/sec/doxm {"oxmsel":0XFF00}
2 RSP 2.04
3 Do vendor specific owner transfer method
DOTS
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)

Figure 19 – Vendor-specific Owner Transfer sequence

Table 5 – Vendor-specific Owner Transfer details

Step	Description
1, 2	The DOTS selects a vendor-specific OTM.
3	The vendor-specific OTM is applied

7.3.7.3 Security considerations

The vendor is responsible for considering security threats and mitigation strategies.

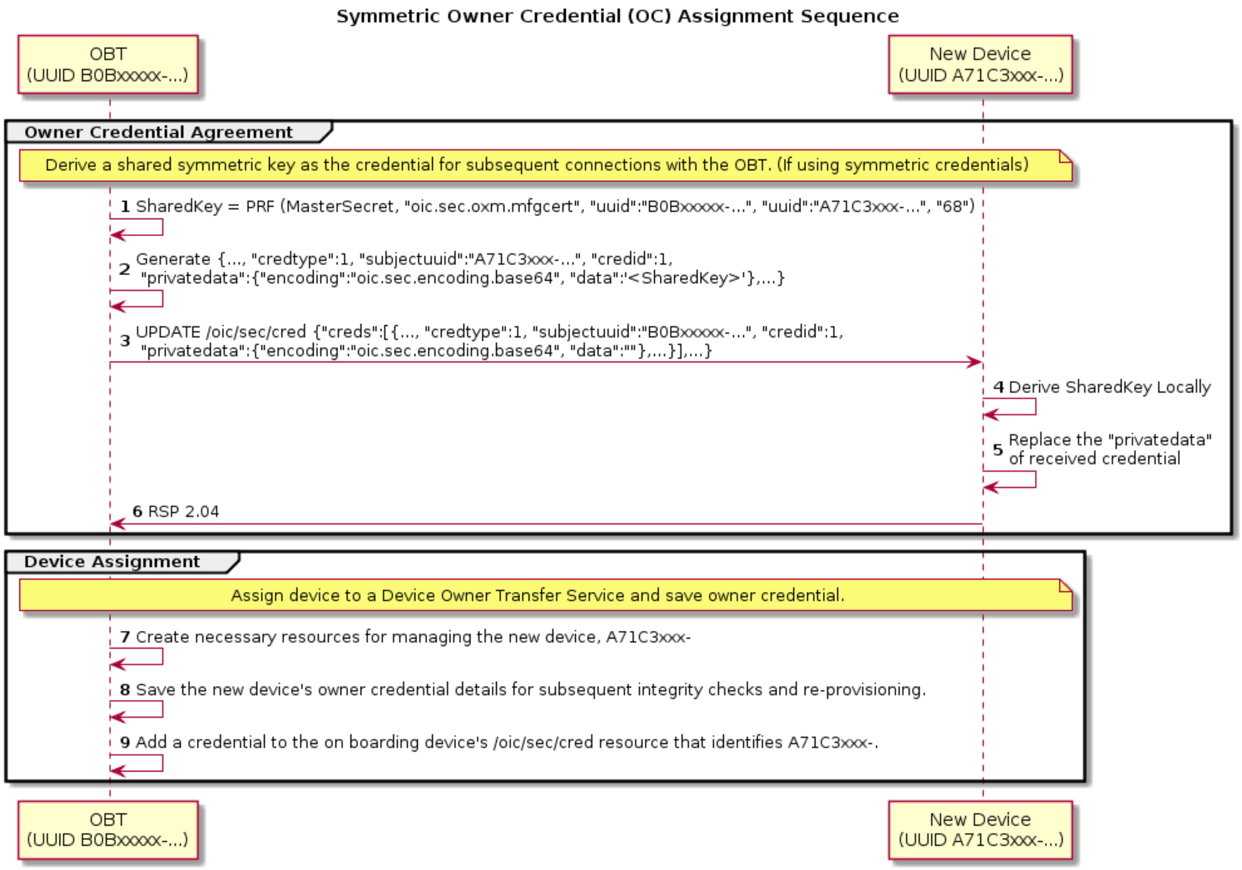
7.3.8 Establishing Owner Credentials

Once the OBT and the new Device have authenticated and established an encrypted connection using one of the defined OTM methods, the Owner Credential(s) can be provisioned.

The Owner Credential is provisioned as part of Ownership Transfer Method, and may be provisioned directly by CMS.

The steps for establishing Device's owner credentials (OC) as part of OTM are:

1) The OBT establishes the Device UUID and Device Owner Id.
2) The OBT then establishes Device’s symmetric OC - See Figure 20 and Table 6.
3) Configure Device services.
4) Configure Device for peer to peer interaction.


Symmetric Owner Credential (OC) Assignment Sequence
OBT
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)
Owner Credential Agreement
Derive a shared symmetric key as the credential for subsequent connections with the OBT. (If using symmetric credentials)
1 SharedKey = PRF (MasterSecret, "oic.sec.oxm.mfgcert", "uuid":"B0Bxxxxx-...", "uuid":"A71C3xxx-...", "68")
2 Generate {..., "credtype":1, "subjectuuid":"A71C3xxx-...", "credid":1,
"privatedata":{"encoding":"oic.sec.encoding.base64", "data":'<SharedKey>'},...}
3 UPDATE /oic/sec/cred {"creds":[{..., "credtype":1, "subjectuuid":"B0Bxxxxx-...", "credid":1,
"privatedata":{"encoding":"oic.sec.encoding.base64", "data":""},...}],...}
4 Derive SharedKey Locally
5 Replace the "privatedata"
of received credential
6 RSP 2.04
Device Assignment
Assign device to a Device Owner Transfer Service and save owner credential.
7 Create necessary resources for managing the new device, A71C3xxx-
8 Save the new device's owner credential details for subsequent integrity checks and re-provisioning.
9 Add a credential to the on boarding device's /oic/sec/cred resource that identifies A71C3xxx-.
OBT
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)

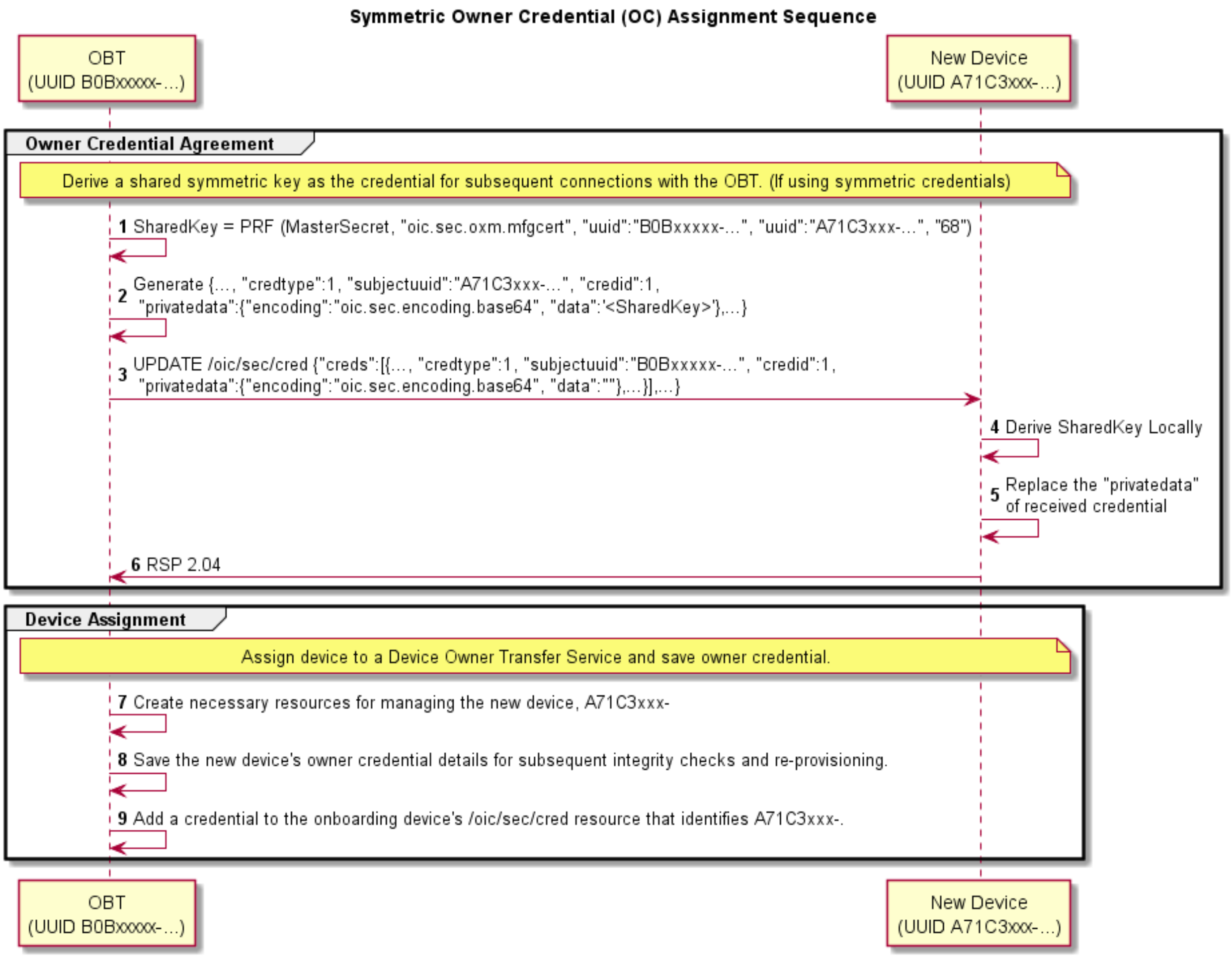

Symmetric Owner Credential (OC) Assignment Sequence
OBT
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)
Owner Credential Agreement
Derive a shared symmetric key as the credential for subsequent connections with the OBT. (If using symmetric credentials)
1 SharedKey = PRF (MasterSecret, "oic.sec.oxm.mfgcert", "uuid":"B0Bxxxxx-...", "uuid":"A71C3xxx-...", "68")
2 Generate {..., "credtype":1, "subjectuuid":"A71C3xxx-...", "credid":1, "privatedata":{"encoding":"oic.sec.encoding.base64", "data":'<SharedKey>'},...}
3 UPDATE /oic/sec/cred {"creds":[{..., "credtype":1, "subjectuuid":"B0Bxxxxx-...", "credid":1, "privatedata":{"encoding":"oic.sec.encoding.base64", "data":""},...}],...}
4 Derive SharedKey Locally
5 Replace the "privatedata" of received credential
6 RSP 2.04
Device Assignment
Assign device to a Device Owner Transfer Service and save owner credential.
7 Create necessary resources for managing the new device, A71C3xxx-
8 Save the new device's owner credential details for subsequent integrity checks and re-provisioning.
9 Add a credential to the onboarding device's /oic/sec/cred resource that identifies A71C3xxx-.
OBT
(UUID B0Bxxxxx-...)
New Device
(UUID A71C3xxx-...)

Figure 20 – Symmetric Owner Credential provisioning sequence

Table 6 – Symmetric Owner Credential assignment details

Step	Description
1, 2	The OBT uses a pseudo-random-function (PRF), the master secret resulting from the DTLS handshake, and other information to generate a symmetric key credential Resource Property - SharedKey.
3	The OBT creates a credential Resource Property set based on SharedKey and then sends the Resource Property set to the new Device with empty "privatedata" Property value.
4, 5	The new Device locally generates the SharedKey and updates it to the "privatedata" Property of the credential Resource Property set.
6	The new Device sends a success message.
7	The onboarding service creates a subjects Resource for the new device (e.g./A71C3xxx-...)
8	The onboarding service provisions its "/oic/svc/dots/subjects/A71C3xxx-/cred" Resource with

	the owner credential. Credential type is SYMMETRIC KEY.
9	(optional) The onboarding service provisions its own "/oic/sec/cred" Resource with the owner credential for new device. Credential type is SYMMETRIC KEY.

In particular, when OBT establishes symmetric owner credentials as part of OTM sequence:

- The OBT generates a Shared Key using the SharedKey Credential Calculation method described in 7.3.2.
- The OBT sends an empty key to the new Device’s "/oic/sec/cred" Resource, identified as a symmetric pair-wise key. The Subject UUID of the "/oic/sec/cred" entry shall match the Device UUID of the OBT.
- Upon receipt of the OBT’s symmetric owner credential, the new Device shall independently generate the Shared Key using the SharedKey Credential Calculation method described in 7.3.2 and store it with the owner credential.
- The new Device shall use the Shared Key owner credential(s) stored via the "/oic/sec/cred" Resource to authenticate the owner during subsequent connections.

7.3.9 Security profile assignment

OCF Devices may have been evaluated according to an OCF Security Profile. Evaluation results could be accessed from a manufacturer’s certificate, OCF web server or other public repository. The DOTS reviews evaluation results to determine which OCF Security Profiles the OCF Device is authorized to possess and configures the Device with the subset of evaluated security profiles best suited for the OCF Security Domain owner’s intended segmentation strategy.

The OCF Device vendor shall set a manufacturer default value for the "supportedprofiles" Property of the "/oic/sec/sp" Resource to match those approved by OCF’s testing and certification process. The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to one of the values contained in the "supportedprofiles". The manufacturer default value shall be re-asserted when the Device transitions to RESET.

The OCF Device shall only allow the "/oic/sec/sp" Resource to be updated when the Device is in one of the following Device States: RFOTM, RFPRO, SRESET and may not allow any update as directed by a Security Profile.

The DOTS may update the "supportedprofiles" Property of the "/oic/sec/sp" Resource with a subset of the OCF Security Profiles values the Device achieved as part of OCF Conformance testing. The DOTS may locate conformance results by inspecting manufacturer certificates supplied with the OCF Device by selecting the "credusage" Property of the "/oic/sec/cred" Resource having the value of "oic.sec.cred.mfgcert". The DOTS may further locate conformance results by visiting a well-known OCF web site URI corresponding to the ocfCPLAttributes extension fields (clause 9.4.2.2.7). The DOTS may select a subset of Security Profiles (from those evaluated by OCF conformance testing) based on a local policy.

As part of onboarding (while the OTM session is active) the DOTS should configure ACE entries to allow DOTS access subsequent to onboarding.

The DOTS should update the "currentprofile" Property of the "/oic/sec/sp" Resource with the value that most correctly depicts the OCF Security Domain owner’s intended Device deployment strategy.

The CMS may issue role credentials using the Security Profile value (e.g. the "sp-blue-v0 OID") to indicate the OCF Security Domain owner’s intention to segment the OCF Security Domain according to a Security Profile. The CMS retrieves the supportedprofiles Property of the "/oic/sec/sp" Resource to select role names corroborated with the Device’s supported Security Profiles when issuing role credentials.

If the CMS issues role credentials based on a Security Profile, the AMS supplies access control entries that include the role designation(s).

7.4 Provisioning

7.4.1 Provisioning flows

7.4.1.1 Provisioning flows general

As part of onboarding a new Device a secure channel is formed between the new Device and the OBT. Subsequent to the Device ownership status being changed to "owned", there is an opportunity to begin provisioning. The OBT provisions the support services that should be subsequently used to complete Device provisioning and on-going Device management.

The Device employs a Client-directed provisioning strategy. The "/oic/sec/pstat" Resource identifies the provisioning strategy and current provisioning status. The provisioning service should determine which provisioning strategy is most appropriate for the OCF Security Domain. See 13.8 for additional detail.

7.4.1.2 Client-directed provisioning

Client-directed provisioning relies on a provisioning service that identifies Servers in need of provisioning then performs all necessary provisioning duties.

An example of Client-directed provisioning is shown in Figure 21 and steps described in Table 7.

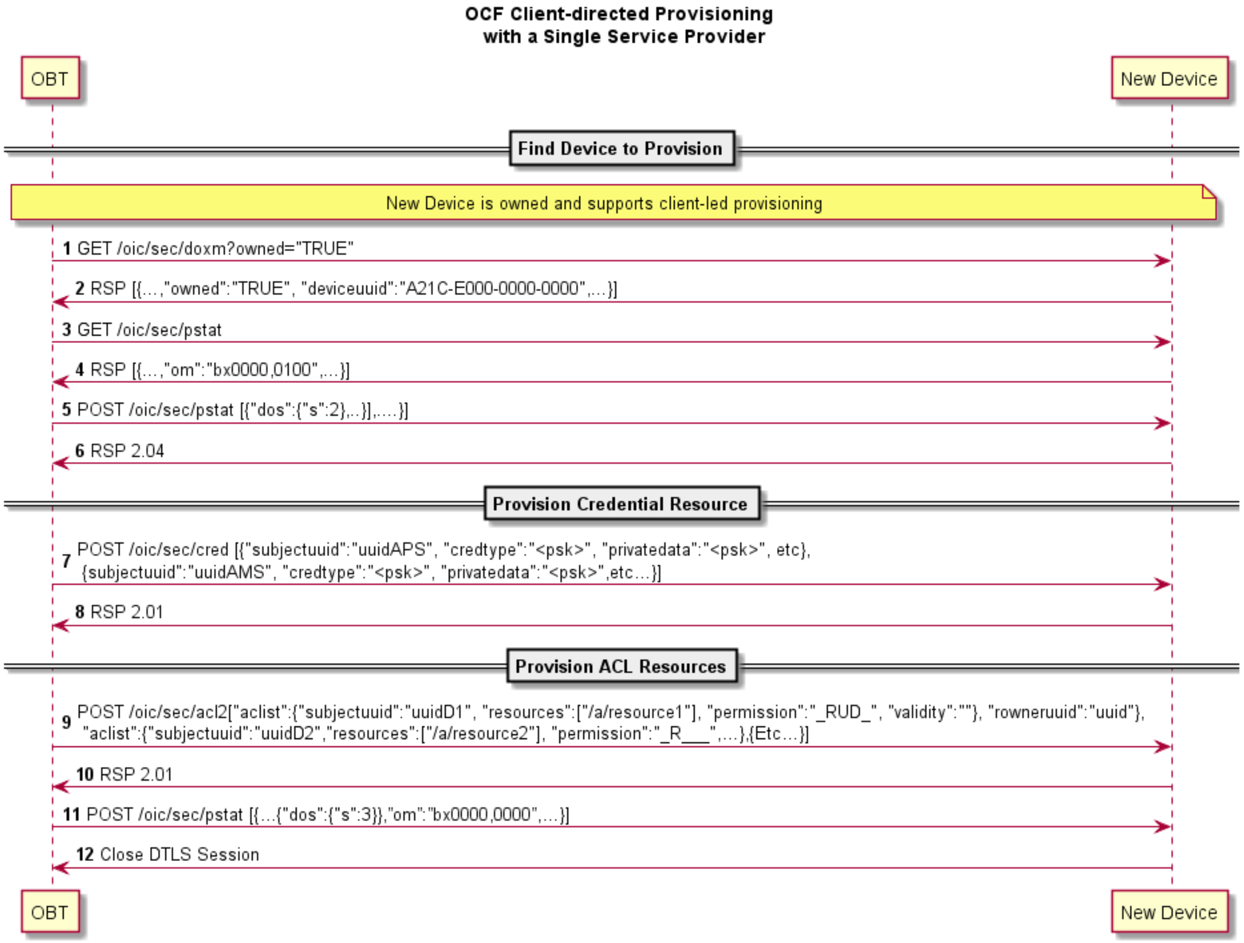

OCF Client-directed Provisioning
with a Single Service Provider
OBT
New Device
Find Device to Provision
New Device is owned and supports client-led provisioning
1 GET /oic/sec/doxm?owned="TRUE"
2 RSP [{…,"owned":"TRUE", "deviceuuid":"A21C-E000-0000-0000",…}]
3 GET /oic/sec/pstat
4 RSP [{…,"om":"bx0000,0100",…}]
5 POST /oic/sec/pstat [{"dos":{"s":2},..}],….}]
6 RSP 2.04
Provision Credential Resource
7 POST /oic/sec/cred [{"subjectuuid":"uuidAPS", "credtype":"<psk>", "privatedata":"<psk>", etc},
{subjectuuid":"uuidAMS", "credtype":"<psk>", "privatedata":"<psk>",etc…}]
8 RSP 2.01
Provision ACL Resources
9 POST /oic/sec/acl2["aclist":{"subjectuuid":"uuidD1", "resources":["/a/resource1"], "permission":"_RUD_", "validity":""}, "rowneruuid":"uuid"},
"aclist":{"subjectuuid":"uuidD2","resources":["/a/resource2"], "permission":"_R___",…},{Etc…}]
10 RSP 2.01
11 POST /oic/sec/pstat [{…{"dos":{"s":3}},"om":"bx0000,0000",…}]
12 Close DTLS Session
OBT
New Device

Figure 21 – Example of Client-directed provisioning

Table 7 – Steps describing Client -directed provisioning

Step	Description
1	Discover Devices that are owned and support Client-directed provisioning.
2	The "/oic/sec/doxm" Resource identifies the Device and it's owned status.
3	DOTS (on OBT) obtains the new Device's provisioning status found in "/oic/sec/pstat" Resource
4	The "pstat" Resource describes the types of provisioning modes supported and which is currently configured. A Device manufacturer should set a default current operational mode ("om"). If the "om" isn't configured for Client-directed provisioning, its "om" value can be changed.
5 - 6	Change Device state to RFPRO.
7 - 8	CMS (on OBT) instantiates the "/oic/sec/cred" Resource. It contains credentials for the provisioned services and other Devices
9 - 10	AMS (on OBT) instantiates "/oic/sec/acl2" Resource.
11	The new Device provisioning status mode is updated to reflect that ACLs have been configured. (RFNOP).
12	The secure session is closed.

7.4.1.3 Server-directed provisioning [DEPRECATED]

This clause is intentionally left blank.

7.4.1.4 Server-directed provisioning Involving multiple support services [DEPRECATED]

This clause is intentionally left blank.

8 Device Onboarding state definitions

8.1 Device Onboarding general

As explained in 5.3, the process of onboarding completes after the ownership of the Device has been transferred and the Device has been provisioned with relevant configuration/services as explained in 5.4. The Figure 22 shows the various states a Device can be in during the Device lifecycle. Device shall reject any requests to perform a state transition not shown on Figure 22.

The "/pstat.dos.s" Property is RW by the "/oic/sec/pstat" Resource owner (e.g. "doxs" service) so that the Resource owner can remotely update the Device state. When the Device is in RFNOP or RFPRO, ACLs can be used to allow remote control of Device state by other Devices. When the Device state is SRESET the Device OC may be the only indication of authorization to access the Device. The Device owner may perform low-level consistency checks and re-provisioning to get the Device suitable for a transition to RFPRO.

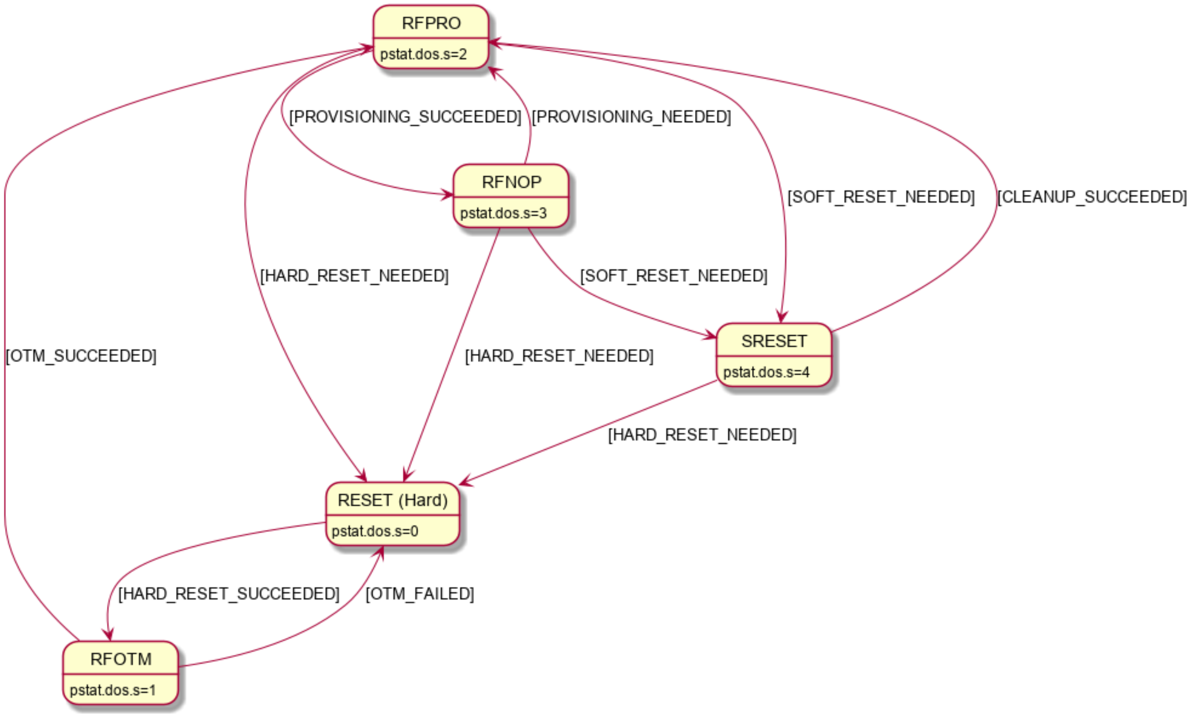


Figure 22 – Device state model

As shown in the diagram, at the conclusion of the provisioning step, the Device comes in RFNOP where it has all it needs in order to start interoperating with other Devices. Clause 8.5 specifies the minimum mandatory configuration that a Device shall hold in order to be considered as RFNOP.

In the event of power loss or Device failure, the Device should remain in the same state that it was in prior to the power loss / failure

If a Device or Resource owner OBSERVEs "/pstat.dos.s", then transitions to SRESET will give early warning notification of Devices that may require SVR consistency checking.

In order for onboarding to function, the Device shall have the following Resources installed:

1) "/oic/sec/doxm" Resource
2) "/oic/sec/pstat" Resource
3) "/oic/sec/cred" Resource

The values contained in these Resources are specified in the state definitions in 8.2, 8.3, 8.4, 8.5 and 8.6. Access policy for these and other SVRs are also described.

8.2 Device Reset state definition

The /pstat.dos.s = RESET is defined as a "hard" reset to manufacturer defaults. Hard reset also defines a state where the Device asset is ready to be transferred to another party.

The Platform manufacturer should provide a physical mechanism (e.g. button) that forces Platform reset. All Devices hosted on the same Platform transition their Device states to RESET when the Platform reset is asserted.

The following Resources and their specific properties shall have the value as specified:

- The "owned" Property of the "/oic/sec/doxm" Resource shall transition to FALSE.
- The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall be nil UUID.

- The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer default value.
- The "sct" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer’s default value.
- The "oxmsel" Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer’s default value.
- The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- The "dos" Property of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal "RESET".
- The "om" (operational modes) Property of the "/oic/sec/pstat" Resource shall be set to the manufacturer default value.
- The "sm" (supported operational modes) Property of the "/oic/sec/pstat" Resource shall be set to the manufacturer default value.
- The "creds" Property of the “/oic/sec/cred” Resource shall be set to the manufacturer default value.
- The "aclist2" Property of the “/oic/sec/acl2” Resource shall be set to the manufacturer default value.
- The "rowneruuid" Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2", and "/oic/sec/cred" Resources shall be nil UUID.
- The "usedspace" Property of the "/oic/sec/ael" Resource shall be set to 0.
- The "categoryfilter" Property of the "/oic/sec/ael" Resource shall be set to the manufacturer's default value.
- The "priorityfilter" Property of the "/oic/sec/ael" Resource shall be set to the manufacturer's default value.
- The "events" Property of the "/oic/sec/ael" Resource shall be set to an empty array.
- The "supportedprofiles" Property of the "/oic/sec/sp" Resource shall be set to the manufacturer default value.
- The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to the manufacturer default value.
- If "/oic/sec/sdi" Resource is exposed by a Device:
  - The "uuid" Property of the Resource shall be set to nil UUID
  - The "name" Property of the Resource shall be set to the empty string
  - The "priv" Property of the Resource shall be set to FALSE
- The Device shall not accept DTLS connection attempts nor TLS connection attempts nor any other requests, including discovery requests.
- Any existing DTLS or TLS Connections shall be closed.

8.3 Device Ready For Owner Tranfer Mechanism state definition

The following Resources and their specific properties shall have the value as specified when the Device enters ready for ownership transfer:

- The "owned" Property of the "/oic/sec/doxm" Resource shall be FALSE and will transition to TRUE.
- The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer default value.
- The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.

- The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFOTM".
- The "/oic/sec/cred" Resource shall contain credential(s) if required by the selected OTM
- If there is no open Device Onboarding Connection, then
  - The Device shall expose an unsecured OCF Endpoint for the Resources "/oic/sec/doxm" and "/oic/sec/pstat".
  - For all SVRs other than "/oic/sec/doxm" and "/oic/sec/pstat":
    - The SVR shall not expose an Unsecured OCF Endpoint.
  - Anonymous Retrieve and Updates requests (those arriving over unauthenticated channel such as CoAP) for the “/oic/sec/doxm” Resource shall be granted.
  - If an anonymous request to Update the “/oic/sec/doxm” Resource attempts to update "oxmsel" to a value that is not indicated as supported by the Device in "oxms", then the Device shall reject the request with an appropriate error message (e.g. bad request).
  - All Retrieve requests to the “/oic/sec/pstat” Resource shall be granted.
  - All other requests, with the exception of Retrieve requests to the Discovery Resources ("/oic/res", "/oic/d" and "/oic/p"), shall be rejected with an appropriate error message (e.g. forbidden).
  - Prior to a successful anonymous Update of "oxmsel" in "/oic/sec/doxm", all attempts to establish new DTLS connections shall be rejected.
  - After a successful anonymous Update of "oxmsel" in "/oic/sec/doxm",
  - The Device shall allow establishing a Device Onboarding Connection (DOC) matching the "oxmsel" Property of the "/oic/sec/doxm" Resource (as specified in clause 7.3) , and shall reject attempts to establish other DTLS connections.
- If there is an open DOC, then
  - For all SVRs:
    - The Device shall not expose an Unsecured OCF Endpoint for the SVR.
  - All requests received over the DOC which target DCRs shall be granted, regardless of the configuration of the ACEs in the "/oic/sec/acl2" Resource.
  - All unicast requests which are not received over the open Device DOC shall be rejected with an appropriate error message (e.g. forbidden), regardless of the configuration of the ACEs in the "/oic/sec/acl2" Resource.
  - All attempts to establish new DTLS connections shall be rejected.
- If the DOC is closed in RFOTM, then the Device shall transition to RESET.

8.4 Device Ready For Provisioning state definition

The following Resources and their specific properties shall have the value as specified when the Device enters ready for provisioning:

- The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be set to the value that was determined during RFOTM processing.
- The "oxmsel" Property of the "/oic/sec/doxm" Resource shall have the value of the actual OTM used during ownership transfer.
- The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFPRO".

- The "rowneruuid" Property of every installed Resource shall be set to a valid Resource owner (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a "rowneruuid" may result in an orphan Resource.
- The "/oic/sec/cred" Resource shall contain credentials for each entity referenced by "rowneruuid" and "devowneruuid" Properties.
- All requests to the "/oic/sec/roles" Resource received over a mutually-authenticated connection established using an identity certificate shall be granted, regardless of the configuration of the ACEs in the "/oic/sec/acl2" Resource, subject to the conditions in clause 10.4.2.
- If there is an open DOC, then all requests received over the DOC which target a DCR shall be granted, regardless of the configuration of the ACEs in the "/oic/sec/acl2" Resource.
- The Device shall allow establishing DTLS connections authenticated with locally issued credentials (clauses 10.2 and 10.4) and shall reject attempts to establish other DTLS connections.
- For all SVRs:
  - The SVR shall not expose an Unsecured OCF Endpoint.
  - The Device shall ignore all ACEs with "subject" matching either {"conntype": "anon-clear"} or {"conntype": "auth-crypt"} when making access decisions for requests to the SVR.

8.5 Device Ready For Normal Operation state definition

The following Resources and their specific properties shall have the value as specified when the Device enters ready for normal operation:

- The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be set to the ID that was configured during OTM. Also the value of the "di" Property in "/oic/d" shall be the same as the deviceuuid.
- The "oxmsel" Property of the "/oic/sec/doxm" Resource shall have the value of the actual OTM used during ownership transfer.
- The "isop" Property of the "/oic/sec/pstat" Resource shall be set to TRUE by the Server once transition to RFNOP is otherwise complete.
- The "dos" of the "/oic/sec/pstat" Resource shall be updated: "dos.s" shall equal "RFNOP".
- The "rowneruuid" Property of every installed Resource shall be set to a valid Resource owner (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a "rowneruuid" results in an orphan Resource.
- The "/oic/sec/cred" Resource shall contain credentials for each service referenced by "rowneruuid" and "devowneruuid" Properties.
- All requests to the "/oic/sec/roles" Resource received over a mutually-authenticated connection established using an identity certificate shall be granted, regardless of the configuration of the ACEs in the "/oic/sec/acl2" Resource, subject to the conditions in clause 10.4.2.
- If there is an open DOC, then requests received over the DOC shall have access decisions determined as follows:
  - A request which targets a DCR shall be granted, regardless of the configuration of the ACEs in the "/oic/sec/acl2" Resource.
  - A request which targets an NCR shall be granted by matching an ACE as per normal request authorization, with "subject" matching the "anon-clear" connection type.
- The Device shall allow establishing DTLS connections authenticated with locally issued credentials and shall reject attempts to establish other DTLS connections.

- For all SVRs:
  - The SVR shall not expose an Unsecured OCF Endpoint.
  - The Device shall ignore all ACEs with "subject" matching either {"conntype": "anon-clear"} or {"conntype": "auth-crypt"} when making access decisions for requests to the SVR.

8.6 Device Soft Reset state definition

The soft reset state is defined (e.g. "/pstat.dos.s" = SRESET) where entrance into this state means the Device is not operational but remains owned by the current owner. The Device may exit SRESET by authenticating to a DOTS (e.g. "rt" = "oic.r.doxs") using the OC provided during original onboarding (but should not require use of an OTM /doxm.oxms).

If the DOTS credential cannot be found or is determined to be corrupted, the Device state transitions to RESET. The Device should remain in SRESET if the DOTS credential fails to validate the DOTS. This mitigates denial-of-service attacks that may be attempted by non-DOTS Devices.

When in SRESET, the following Resources and their specific Properties shall have the values as specified.

- The "owned" Property of the "/oic/sec/doxm" Resource shall be TRUE.
- The "devowneruuid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- The "deviceuuid" Property of the "/oic/sec/doxm" Resource shall remain non-null.
- The "sct" Property of the "/oic/sec/doxm" Resource shall retain its value.
- The "oxmsel" Property of the "/oic/sec/doxm" Resource shall retains its value.
- The "isop" Property of the "/oic/sec/pstat" Resource shall be FALSE.
- The "/oic/sec/pstat.dos.s" Property shall be SRESET.
- The "om" (operational modes) Property of the "/oic/sec/pstat" Resource shall be "client-directed mode".
- The "sm" (supported operational modes) Property of "/oic/sec/pstat" Resource may be updated by the Device owner (aka DOTS).
- The "rowneruuid" Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2", and "/oic/sec/cred" Resources may be reset by the Device owner (aka DOTS) and re-provisioned.
- All requests to the "/oic/sec/roles" Resource received over a mutually-authenticated connection established using an identity certificate shall be granted, regardless of the configuration of the ACEs in the "/oic/sec/acl2" Resource, subject to the conditions in clause 10.4.2.
- If there is an open DOC, then all requests received over the DOC which target a DCR shall be granted, regardless of the configuration of the ACEs in the "/oic/sec/acl2" Resource.
- The Device shall allow establishing DTLS connections authenticated with locally issued credentials and shall reject attempts to establish other DTLS connections.
- For all SVRs:
  - The SVR shall not expose an Unsecured OCF Endpoint.
  - The Device shall ignore all ACEs with "subject" matching either {"conntype": "anon-clear"} or {"conntype": "auth-crypt"} when making access decisions for requests to the SVR.

9 Security Credential management

9.1 Preamble

This clause provides an overview of the credential types in OCF, along with details of credential use, provisioning and ongoing management.

9.2 Credential lifecycle

9.2.1 Credential lifecycle general

OCF credential lifecycle has the following phases: (1) creation, (2) deletion, (3) refresh and (4) revocation.

9.2.2 Creation

The CMS can provision credentials to the credential Resource onto the Device. The Device shall verify the CMS is authorized by matching the rowneruuid Property of the "/oic/sec/cred" Resource to the DeviceID of the credential the CMS used to establish the secure connection.

Credential Resources created using a CMS may involve specialized credential issuance protocols and messages. These may involve the use of public key infrastructure (PKI) such as a certificate authority (CA), symmetric key management such as a key distribution centre (KDC) or as part of a provisioning action by a DOTS, CMS or AMS.

9.2.3 Deletion

The CMS can delete credentials from the credential Resource. The Device (e.g. the Device where the credential Resource is hosted) should delete credential Resources that have expired.

An expired credential Resource may be deleted to manage memory and storage space.

Deletion in OCF key management is equivalent to credential suspension.

9.2.4 Refresh

Credential refresh may be performed before it expires. The CMS performs credential refresh.

The "/oic/sec/cred" Resource supports expiry using the Period Property. Credential refresh may be applied when a credential is about to expire or is about to exceed a maximum threshold for bytes encrypted.

A credential refresh method specifies the options available when performing key refresh. The Period Property informs when the credential should expire. The Device may proactively obtain a new credential using a credential refresh method using current unexpired credentials to refresh the existing credential. If the Device does not have an internal time source, the current time should be obtained from a CMS at regular intervals.

If the onboarding established credentials are allowed to expire the DOTS shall re-onboard the Device to re-apply device owner transfer steps.

All Devices shall support at least one credential refresh method.

9.2.5 Revocation

Credentials issued by a CMS may be equipped with revocation capabilities. In situations where the revocation method involves provisioning of a revocation object that identifies a credential that is to be revoked prior to its normal expiration period, a credential Resource is created containing the revocation information that supersedes the originally issued credential. The revocation object expiration should match that of the revoked credential so that the revocation object is cleaned up upon expiry.

It is conceptually reasonable to consider revocation applying to a credential or to a Device. Device revocation asserts all credentials associated with the revoked Device should be considered for revocation. Device revocation is necessary when a Device is lost, stolen or compromised. Deletion of credentials on a revoked Device might not be possible or reliable.

9.3 Credential types

9.3.1 Preamble

The "/oic/sec/cred" Resource maintains a credential type Property that supports several cryptographic keys and other information used for authentication and data protection. The credential types supported include symmetric pair-wise key, group symmetric group key, asymmetric signing key, asymmetric signing key with certificate and shared-secret (i.e. PIN or password). The Device shall always support symmetric pair-wise key and asymmetric signing key with certificate credential types. Other credential types are optional.

9.3.2 Pair-wise symmetric key credentials

The CMS shall provision exactly one other pair-wise symmetric credential to a peer Device. The CMS should not store pair-wise symmetric keys it provisions to managed Devices.

Pair-wise keys could be established through ad-hoc key agreement protocols.

The "PrivateData" Property in the "/oic/sec/cred" Resource contains the symmetric key.

The "PublicData" Property may contain a token encrypted to the peer Device containing the pair-wise key.

The "OptionalData" Property may contain revocation status.

The Device implementer should apply hardened key storage techniques that ensure the "PrivateData" remains private.

The Device implementer should apply appropriate integrity, confidentiality and access protection of the "/oic/sec/cred", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent unauthorized modifications.

9.3.3 Group symmetric key credentials

Group keys are symmetric keys shared among a group of Devices (3 or more). Group keys are used for efficient sharing of data among group participants.

Group keys do not provide authentication of Devices but only establish membership in a group.

The CMS shall provision group symmetric key credentials to the group members. The CMS maintains the group memberships.

The "PrivateData" Property in the "/oic/sec/cred" Resource contains the symmetric key.

The "PublicData" Property may contain the group name.

The "OptionalData" Property may contain revocation status.

The Device implementer should apply hardened key storage techniques that ensure the "PrivateData" remains private.

The Device implementer should apply appropriate integrity, confidentiality and access protection of the "/oic/sec/cred", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent unauthorized modifications.

9.3.4 Asymmetric authentication key credentials

9.3.4.1 Asymmetric authentication key credentials general

Asymmetric authentication key credentials contain either a public and private key pair or only a public key. The private key is used to sign Device authentication challenges. The public key is used to verify a device authentication challenge-response.

The "PrivateData" Property in the "/oic/sec/cred" Resource contains the private key.

The "PublicData" Property contains the public key.

The "OptionalData" Property may contain revocation status.

The Device implementer should apply hardened key storage techniques that ensure the "PrivateData" remains private.

Devices should generate asymmetric authentication key pairs internally to ensure the private key is only known by the Device. See 9.3.4.2 for when it is necessary to transport private key material between Devices.

The Device implementer should apply appropriate integrity, confidentiality and access protection of the "/oic/sec/cred", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent unauthorized modifications.

9.3.4.2 External creation of asymmetric authentication key credentials

Devices should employ industry-standard high-assurance techniques when allowing off-device key pair creation and provisioning. Use of such key pairs should be minimized, particularly if the key pair is immutable and cannot be changed or replaced after provisioning.

When used as part of onboarding, these key pairs can be used to prove the Device possesses the manufacturer-asserted properties in a certificate to convince a DOTS or a user to accept onboarding the Device. See 7.3.3 for the OTM that uses such a certificate to authenticate the Device, and then provisions new OCF Security Domain credentials for use.

9.3.5 Asymmetric key encryption key credentials

The asymmetric key-encryption-key (KEK) credentials are used to wrap symmetric keys when distributing or storing the key.

The "PrivateData" Property in the "/oic/sec/cred" Resource contains the private key.

The "PublicData" Property contains the public key.

The "OptionalData" Property may contain revocation status.

The Device implementer should apply hardened key storage techniques that ensure the "PrivateData" remains private.

The Device implementer should apply appropriate integrity, confidentiality and access protection of the "/oic/sec/cred", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent unauthorized modifications.

9.3.6 Certificate credentials

Certificate credentials are asymmetric keys that are accompanied by a certificate issued by a CMS or an external certificate authority (CA).

A certificate enrolment protocol is used to obtain a certificate and establish proof-of-possession.

The issued certificate is stored with the asymmetric key credential Resource.

Other objects useful in managing certificate lifecycle such as certificate revocation status are associated with the credential Resource.

Either an asymmetric key credential Resource or a self-signed certificate credential is used to terminate a path validation.

The "PrivateData" Property in the "/oic/sec/cred" Resource contains the private key.

The "PublicData" Property contains the issued certificate.

The "OptionalData" Property may contain revocation status.

The Device implementer should apply hardened key storage techniques that ensure the PrivateData remains private.

The Device implementer should apply appropriate integrity, confidentiality and access protection of the "/oic/sec/cred", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent unauthorized modifications.

9.3.7 Password credentials

The "PrivateData" Property in the "/oic/sec/cred" Resource contains the PIN, password and other values useful for changing and verifying the password.

The "PublicData" Property may contain the user or account name if applicable.

The "OptionalData" Property may contain revocation status.

The Device implementer should apply hardened key storage techniques that ensure the "PrivateData" remains private.

The Device implementer should apply appropriate integrity, confidentiality and access protection of the "/oic/sec/cred", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent unauthorized modifications.

9.3.8 Credentials for direct provisioning an OSCORE security context

A credential entry with the credential type 64 is used for direct provisioning of OSCORE Security Context parameters for use in End-to-End Security of Unicast Messages.

The "privatedata" Property of the credential entry with the credential type 64 in the "/oic/sec/cred" Resource contains the OSCORE Master Key.

A credential entry with the credential type 64 shall expose the OSCORE Configuration ("oscore") Property, which includes:

– The "senderid" Property containing the OSCORE Sender ID parameter.

– The "recipientid" Property containing the OSCORE Recipient ID parameter.

– The "ssn" Property contains a read-only value used to store the OSCORE Sender Sequence Number.

NOTE: values of "senderid" and "recipientid" are expected to be lowercase hexadecimal encoded with "0x" encoding prefix omitted.

See clause16.2 for description of the OSCORE parameters.

9.3.9 Credentials for Simple Secure Multicast

There are two distinct credential types used for provisioning OSCORE Security Context parameters used in Simple Secure Multicast (SSM): one for the SSM Client Context identified using "credtype" :"128"; and one for the SSM Server Context identified using "credtype" : "256". In a

Client of an SSM Group, the Client's OSCORE Security Context (Sender context) is derived from a provisioned SSM Client Context. In the Servers of an SSM Group, the Server's OSCORE Security Context (Recipient Context) is derived from a provisioned SSM Server Context.

For both of these credential types, the "privatedata" Property of the credential entry in the "/oic/sec/cred" Resource contains the value of the OSCORE Master Secret of the SSM Group, which is generated by the OBT.

A SSM Client Context credential entry shall expose the OSCORE Configuration ("oscore") Property, which for this credential type shall include:

– The "senderid" Property containing the OSCORE Sender ID parameter.
    – This value is selected and provisioned by the OBT.
– The “desc” Property containing a description of the usage of the security context
    – This Property contains a human-readable description intended for identifying the corresponding SSM Group when a Security Domain contains multiple SSM Groups.
    – This value is selected and provisioned by the OBT
– The "ssn" Property contains a read-only value used to store the OSCORE Sender Sequence Number.

NOTE 1: The value of "senderid" is expected to be lowercase hexadecimal encoded with "0x" encoding prefix omitted.

An SSM Server Context credential entry shall include the OSCORE Configuration ("oscore") Property, which shall include:

– The "recipientid" Property containing the OSCORE Group Recipient ID parameter.
    – This value is equal for all Servers in the SSM Group, and is the same as the value of the "senderid" of the Client Context for the SSM Group
    – This value is selected and provisioned by the OBT
– The “desc” Property containing a description of the usage of the security context
    – This Property contains a human-readable description intended for identifying the corresponding SSM Group when a Security Domain contains multiple SSM Groups.
    – This value is selected and provisioned by the OBT

NOTE 2: The value of "recipientid" is expected to be lowercase hexadecimal encoded with "0x" encoding prefix omitted.

See clause 16.3.3 for description of the OSCORE parameters used in SSM.

9.4 Certificate based key management

9.4.1 Overview

To achieve authentication and transport security during communications in OCF Security Domain, certificates containing public keys of communicating parties and private keys can be used.

The certificate and private key may be issued by a local or remote certificate authority (CA).

The OCF certificate format is a subset of X.509 format, only elliptic curve algorithm and PEM encoding format are allowed, most of optional fields in X.509 are not supported so that the format intends to meet the constrained Device’s requirement.

The CMS manages the certificate lifecycle for certificates it issues. The DOTS assigns a CMS to a Device when it is newly onboarded.

9.4.2 X.509 Digital certificate profiles

9.4.2.1 Digital certificate profile general

An OCF certificate format is a subset of X.509 format (version 3 or above) as defined in IETF RFC 5280.

This clause develops a profile to facilitate the use of X.509 certificates within OCF applications for those communities wishing to make use of X.509 technology. The X.509 v3 certificate format is described in detail, with additional information regarding the format and semantics of OCF specific extension(s). The supported standard certificate extensions are also listed.

Certificate Format: The OCF certificate profile is derived from IETF RFC 5280. However, this document does not support the "issuerUniqueID" and "subjectUniqueID" fields which are deprecated and shall not be used in the context of OCF. If these fields are present in a certificate, compliant entities shall ignore their contents.

Certificate Encoding: Conforming entities shall use the Privacy-Enhanced Mail (PEM) to encode certificates.

Certificates Hierarchy and Crypto Parameters. OCF supports a three-tier hierarchy for its Public Key Infrastructure (i.e., a Root CA, an Intermediate CA, and EE certificates). OCF accredited CAs SHALL use Elliptic Curve Cryptography (ECC) keys (secp256r1 – OID:1.2.840.10045.3.1.7) and use the ecdsaWithSHA256 (OID:1.2.840.10045.4.3.2) algorithm for certificate signatures. Elliptic Curve Cryptography public keys shall be encoded using uncompressed Elliptic Curve points.

The following clauses specify the supported standard and custom extensions for the OCF certificates profile.

9.4.2.2 Certificate profile and fields

9.4.2.2.1 Root CA certificate profile

Table 8 describes X.509 v1 fields required for Root CA Certificates.

Table 8 – X.509 v1 fields for Root CA certificates

V1 Field	Value / Remarks
signatureAlgorithm	ecdsa-with-SHA256 (OID: 1.2.840.10045.4.3.2)
Version	v3 (value is 2)
SerialNumber	SHALL be a positive integer, unique among all certificates issued by a given CA
Issuer	SHALL match the Subject field
Subject	SHALL match the Issuer field
notBefore	The time at which the Root CA Certificate was generated. See 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
notAfter	No stipulation for expiry date. See 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
Subject Public Key Info	id-ecPublicKey (OID: 1.2.840.10045.2.1) secp256r1 (OID:1.2.840.10045.3.1.7) Elliptic Curve Cryptography public keys shall be encoded using uncompressed Elliptic Curve points.

Table 9 describes X.509 v3 extensions required for Root CA Certificates.

Table 9 - X.509 v3 extensions for Root CA certificates

Extension	Required/Optional	Criticality	Value / Remarks
authorityKeyIdentifier	OPTIONAL	Non-critical	N/A
subjectKeyIdentifier	OPTIONAL	Non-critical	N/A
keyUsage	REQUIRED	Critical	keyCertSign (5) & cRLSign (6) bits shall be enabled. digitalSignature(0) bit may be enabled. All other bits shall not be enabled.

basicConstraints	REQUIRED	Critical	cA = TRUE
			pathLenConstraint = not present (unlimited)

9.4.2.2.2 Intermediate CA certificate profile

Table 10 describes X.509 v1 fields required for intermediate CA certificates.

Table 10 - X.509 v1 fields for intermediate CA certificates

V1 Field	Value / Remarks
signatureAlgorithm	ecdsa-with-SHA256 (OID: 1.2.840.10045.4.3.2)
Version	v3 (value is 2)
SerialNumber	SHALL be a positive integer, unique among all certificates issued by Root CA
Issuer	SHALL match the Subject field of the issuing Root CA
Subject	(no stipulation)
notBefore	The time at which the Intermediate CA Certificate was generated. See clause 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
notAfter	No stipulation for expiry date. See clause10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
Subject Public Key Info	id-ecPublicKey (OID: 1.2.840.10045.2.1) secp256r1 (OID:1.2.840.10045.3.1.7) Elliptic Curve Cryptography public keys shall be encoded using uncompressed Elliptic Curve points.

Table 11 describes X.509 v3 extensions required for intermediate CA certificates.

Table 11 – X.509 v3 extensions for intermediate CA certificates

Extension	Required/Optional	Criticality	Value / Remarks
authorityKeyIdentifier	OPTIONAL	Non-critical	N/A
subjectKeyIdentifier	OPTIONAL	Non-critical	N/A
keyUsage	REQUIRED	Critical	keyCertSign (5) & cRLSign (6) bits shall be enabled. digitalSignature (0) bit may be enabled All other bits shall not be enabled.
basicConstraints	REQUIRED	Critical	cA = TRUE

			pathLenConstraint = 0 (can only sign End-Entity certs)
certificatePolicies	OPTIONAL	Non-critical	(no stipulation)
cRLDistributionPoints	OPTIONAL	Non-critical	1 or more URIs where the Certificate Revocation List (CRL) from the Root can be obtained.
authorityInformationAccess	OPTIONAL	Non-critical	OCSP URI – the URI of the Root CA’s OCSP Responder

9.4.2.2.3 End-Entity Black certificate profile

Table 12 describes X.509 v1 fields required for end-entity eertificates used for Black security profile.

Table 12 – X.509 v1 fields for end-entity certificates

V1 Field	Value / Remarks
signatureAlgorithm	ecdsa-with-SHA256 (OID: 1.2.840.10045.4.3.2)
Version	v3 (value is 2)
SerialNumber	SHALL be a positive integer, unique among all certificates issued by the Intermediate CA
Issuer	SHALL match the Subject field of the issuing Intermediate CA
Subject	Subject DN shall include: o=OCF-verified device manufacturer organization name. The Subject DN may include other attributes (e.g. cn, c, ou, etc.) with no stipulation by OCF.
notBefore	The time at which the End-Entity Certificate was generated. See clause 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
notAfter	No stipulation. See clause 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
Subject Public Key Info	id-ecPublicKey (OID: 1.2.840.10045.2.1) secp256r1 (OID:1.2.840.10045.3.1.7) Elliptic Curve Cryptography public keys shall be encoded using uncompressed Elliptic Curve points.

Table 13 describes X.509 v3 extensions required for end-entity certificates.

Table 13 – X.509 v3 extensions for end-entity Certificates

Extension	Required/ Optional	Criticality	Value / Remarks
authorityKeyIdentifier	OPTIONAL	Non-critical	N/A
subjectKeyIdentifier	OPTIONAL	Non-critical	N/A
keyUsage	REQUIRED	Critical	digitalSignature (0) and keyAgreement(4) bits SHALL be the only bits enabled

basicConstraints	OPTIONAL	Non-Critical	cA = FALSE
			pathLenConstraint = not present

certificatePolicies	OPTIONAL	Non-critical	End-Entity certificates chaining to an OCF Root CA SHOULD contain at least one PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2) corresponding to the version of the OCF Certificate Policy under which it was issued. Additional manufacturer-specific CP OIDs may also be populated.
extendedKeyUsage	REQUIRED	Non-critical	The following extendedKeyUsage (EKU) OIDs SHALL both be present: • serverAuthentication - 1.3.6.1.5.5.7.3.1 • clientAuthentication - 1.3.6.1.5.5.7.3.2 Exactly ONE of the following OIDs SHALL be present: • Identity certificate - 1.3.6.1.4.1.44924.1.6 • Role certificate - 1.3.6.1.4.1.44924.1.7 End-Entity certificates SHALL NOT contain the anyExtendedKeyUsage OID (2.5.29.37.0)
subjectAlternativeName	REQUIRED UNDER CERTAIN CONDITIONS	Non-critical	The subjectAltName extension is used to encode one or more Role ID values in role certificates, binding the roles to the subject public key. When the extendedKeyUsage (EKU) extension contains the Identity Certificate OID (1.3.6.1.4.1.44924.1.6), the subjectAltName extension SHOULD NOT be present. If the EKU extension contains the Role Certificate OID (1.3.6.1.4.1.44924.1.7), the subjectAltName extension SHALL be present and populated as follows: Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name shall contain exactly one CN (Common Name) component, and zero or one OU (Organizational Unit) components. The OU component, if present, shall specify the authority that defined the semantics of the

			role. If the OU component is absent, the certificate issuer has defined the role. The CN component shall encode the role ID. Other GeneralName types in the SEQUENCE may be present, but shall not be interpreted as roles. The role, and authority shall be encoded as ASN.1 PrintableString type, the restricted character set [0-9a-z-A-z '()+,-./:=?].
cRLDistributionPoints	OPTIONAL	Non-critical	1 or more URIs where the Certificate Revocation List (CRL) from the Intermediate CA can be obtained.
authorityInformationAccess	OPTIONAL	Non-critical	OCSP URI – the URI of the Intermediate CA's OCSP Responder
OCF Compliance	OPTIONAL	Non-critical	See 9.4.2.2.4
Manufacturer Usage Description (MUD)	OPTIONAL	Non-critical	Contains a single Uniform Resource Locator (URL) that points to an on-line Manufacturer Usage Description concerning the certificate subject. See 9.4.2.2.5
OCF Security Claims	OPTIONAL	Non-critical	Contains a list of security claims above those required by this OCF Compliance version or Security Profile. See 9.4.2.2.6
OCF CPL Attributes	OPTIONAL	Non-critical	Contains the list of OCF Attributes used to perform OCF Certified Product List lookups

9.4.2.2.4 OCF Compliance X.509v3 Extension

The OCF Compliance Extension defines required parameters to correctly identify the type of Device, its manufacturer, its OCF Version, and the Security Profile compliance of the device.

The extension carries an "ocfVersion" field which provides the specific base version of the OCF documents the device implements. The "ocfVersion" field shall contain a sequence of three integers ("major", "minor", and "build"). For example, if an entity is certified to be compliant with OCF specifications 1.3.2, then the "major", "minor", and "build" fields of the "ocfVersion" will be set to "1", "3", and "2" respectively. The "ocfVersion" may be used by Security Profiles to denote compliance to a specified base version of the OCF documents.

The "securityProfile" field shall carry the ocfSecurityProfile OID(s) (clause 14.8.3) of one or more supported Security Profiles associated with the certificate in string form (UTF-8). All Security Profiles associated with the certificate should be identified by this field.

The extension shall also carry two string fields (UTF-8): "DeviceName" and "deviceManufacturer". The fields carry human-readable descriptions of the Device's name and manufacturer, respectively.

The ASN.1 definition of the OCFCompliance extension (OID – 1.3.6.1.4.1.51414.1.0) is defined as follows:

```
id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
                               private(4) enterprise(1) OCF(51414) }

```

```
  id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }

    id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }

ocfVersion ::= SEQUENCE {
       major  INTEGER,
              --Major version number
       minor  INTEGER,
              --Minor version number
       build  INTEGER,
              --Build/Micro version number
}

ocfCompliance ::= SEQUENCE {
       version                    ocfVersion,
                           --Device/OCF version
       securityProfile            SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
                           --Sequence of OCF Security Profile OID strings
                                  --Clause 14.8.2 defines valid ocfSecurityProfileOIDs
       deviceName          UTF8String,
                           --Name of the device
       deviceManufacturer  UTF8String,
                           --Human-Readable Manufacturer
                           --of the device
}
```

9.4.2.2.5 Manufacturer Usage Description (MUD) X.509v3 Extension

The goal of the Manufacturer Usage Description (MUD) extension is to provide a means for devices to signal to the network the access and network functionality they require to properly function. Access controls can be more easily achieved and deployed at scale when the MUD extension is used.

The MUD X.509 v3 extension is specified in IETF RFC 8520 with the full ASN.1 definition in clause 11.

9.4.2.2.6 OCF Security Claims X.509v3 Extension

The OCF Security Claims Extension defines a list of OIDs representing security claims that the manufacturer/integrator is making as to the security posture of the device above those required by the OCF Compliance version or that of the OCF Security Profile being indicated by the device.

The purpose of this extension is to allow for programmatic evaluation of assertions made about security to enable some platforms/policies/administrators to better understand what is being onboarded or challenged.

The ASN.1 definition of the OCF Security Claims extension (OID – 1.3.6.1.4.1.51414.1.1) is defined as follows:

```
id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
                                      private(4) enterprise(1) OCF(51414) }

    id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }

    id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }

        claim-secure-boot            ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
        --Device claims that the boot process follows a procedure trusted
        --by the firmware and the BIOS

        claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
        --Device claims that credentials are stored in a specialized hardware
        --protection environment such as a Trusted Platform Module (TPM) or
        --similar mechanism.

          ocfSecurityClaimsOID ::= OBJECT IDENTIFIER

    ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID
```

9.4.2.2.7 OCF Certified Product List Attributes X.509v3 Extension

The OCF Certified Product List Extension defines required parameters to utilize the OCF Compliance Management System Certified Product List (OCMS-CPL). This clause is only applicable if you plan to utilize the OCMS-CPL. The OBT may make use of these attributes to verify the compliance level of a device.

The extension carries the OCF CPL Attributes: IANA Private Enterprise Number (PEN), Model and Version.

The 'cpl-at-IANAPen' IANA Private Enterprise Number (PEN) provides the manufacturer's unique PEN established in the IANA PEN list located at: https://www.iana.org/assignments/enterprise-numbers. The 'cpl-at-IANAPen' field found in end-products shall be the same information as reported during OCF Certification.

The 'cpl-at-model' represents an OCF-Certified product's model name. The 'cpl-at-model' field found in end-products shall be the same information as reported during OCF Certification.

The 'cpl-at-version' represents an OCF-Certified product's version. The 'cpl-at-version' field found in end-products shall be the same information as reported during OCF Certification.

The ASN.1 definition of the OCF CPL Attributes extension (OID – 1.3.6.1.4.1.51414.1.2) is defined as follows:

```
id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
                                      private(4) enterprise(1) OCF(51414) }

id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }

    id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }

      cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
      cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
      cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }


  ocfCPLAttributes ::= SEQUENCE {
       cpl-at-IANAPen      UTF8String,
                    --Manufacturer's registered IANA Private Enterprise Number
       cpl-at-model        UTF8String,
                    --Device OCF Security Profile
       cpl-at-version      UTF8String
                    --Name of the device
}
```

9.4.2.3 Supported certificate extensions

As these certificate extensions are a standard part of IETF RFC 5280, this document includes the clause number from that RFC to include it by reference. Each extension is summarized here, and any modifications to the RFC definition are listed. Devices shall be capable of parsing and enforcing the extensions listed here; other extensions from the RFC are not included in this document and therefore are not required. Clause 10.4 describes what Devices implement when validating certificate chains, including processing of extensions, and actions to take when certain extensions are absent.

– Authority Key Identifier (4.2.1.1)

The Authority Key Identifier (AKI) extension provides a means of identifying the public key corresponding to the private key used to sign a certificate. This document makes the following modifications to the referenced definition of this extension:

The "authorityCertIssuer" or "authorityCertSerialNumber" fields of the "AuthorityKeyIdentifier" sequence are not permitted; only "keyIdentifier" is allowed. This results in the following grammar definition:

```
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 35 }

AuthorityKeyIdentifier ::= SEQUENCE {
      keyIdentifier             [0] KeyIdentifier           }

KeyIdentifier ::= OCTET STRING
```

– Subject Key Identifier (4.2.1.2)

The Subject Key Identifier (SKI) extension provides a means of identifying certificates that contain a particular public key.

This document makes the following modification to the referenced definition of this extension:

Subject Key Identifiers should be derived from the public key contained in the certificate’s "SubjectPublicKeyInfo" field or a method that generates unique values. This document RECOMMENDS the 256-bit SHA-2 hash of the value of the BIT STRING "subjectPublicKey" (excluding the tag, length, and number of unused bits). Devices verifying certificate chains shall not assume any particular method of computing key identifiers, and shall only base matching AKI’s and SKI’s in certification path constructions on key identifiers seen in certificates.

– Subject Alternative Name

If the EKU extension is present, and has the Role Certificate OID (1.3.6.1.4.1.44924.1.7) , indicating that this is a role certificate, the Subject Alternative Name (subjectAltName) extension shall be present and interpreted as described below. When no EKU is present, or has another value, the "subjectAltName" extension should be absent. The "subjectAltName" extension is used to encode one or more Role ID values in role certificates, binding the roles to the subject public key. The "subjectAltName" extension is defined in IETF RFC 5280 (See 4.2.1.6):

```
id-ce-subjectAltName OBJECT IDENTIFIER ::=  { id-ce 17 }

SubjectAltName ::= GeneralNames

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
        otherName                       [0]     OtherName,
        rfc5322Name                     [1]     IA5String,
        dNSName                         [2]     IA5String,
        x400Address                     [3]     ORAddress,
        directoryName                   [4]     Name,
        ediPartyName                    [5]     EDIPartyName,
        uniformResourceIdentifier       [6]     IA5String,
        iPAddress                       [7]     OCTET STRING,
        registeredID                    [8]     OBJECT IDENTIFIER }

      EDIPartyName ::= SEQUENCE {
        nameAssigner            [0]     DirectoryString OPTIONAL,
        partyName               [1]     DirectoryString }
```

Each "GeneralName" in the "GeneralNames" SEQUENCE which encodes a role shall be a "directoryName", which is of type Name. Name is an X.501 Distinguished Name. Each Name shall contain exactly one CN (Common Name) component, and zero or one OU (Organizational Unit) components. The OU component, if present, shall specify the authority that defined the semantics of the role. If the OU component is absent, the certificate issuer has defined the role. The CN component shall encode the role ID. Other "GeneralName" types in the SEQUENCE may be present, but shall not be interpreted as roles. Therefore, if the certificate issuer includes non-role names in the "subjectAltName" extension, the extension should not be marked critical.

The role, and authority need to be encoded as ASN.1 "PrintableString" type, the restricted character set [0-9a-z-A-z '()+,-./:=?].

– Key Usage (4.2.1.3)

The key usage extension defines the purpose (e.g., encipherment, signature, certificate signing) of the key contained in the certificate. The usage restriction might be employed when a key that could be used for more than one operation is to be restricted.

This document does not modify the referenced definition of this extension.

– Basic Constraints (4.2.1.9)

The basic constraints extension identifies whether the subject of the certificate is a CA and the maximum depth of valid certification paths that include this certificate. Without this extension, a certificate cannot be an issuer of other certificates.

This document does not modify the referenced definition of this extension.

– Extended Key Usage (4.2.1.12)

Extended Key Usage describes allowed purposes for which the certified public key may be used. When a Device receives a certificate, it determines the purpose based on the context of the interaction in which the certificate is presented, and verifies the certificate may be used for that purpose.

This document makes the following modifications to the referenced definition of this extension:

CAs should mark this extension as critical.

CAs shall not issue certificates with the anyExtendedKeyUsage OID (2.5.29.37.0).

The list of OCF-specific purposes and the assigned OIDs to represent them are:

– Identity certificate	1.3.6.1.4.1.44924.1.6
– Role certificate	1.3.6.1.4.1.44924.1.7

9.4.2.4 Cipher suite for authentication, confidentiality, and integrity

OCF compliant entities shall support TLS version 1.2. Compliant entities shall support TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suite as defined in IETF RFC 7251 and may support additional ciphers as defined in the TLS v1.2 specifications.

9.4.2.5 Encoding of certificate

See 9.4.2 for details.

9.4.3 Certificate Revocation List (CRL) Profile [Deprecated]

This clause is intentionally left blank.

9.4.4 Resource model

Device certificates and private keys are kept in "cred" Resource.

The "cred" Resource contains the certificate information pertaining to the Device. The "PublicData" Property holds the device certificate and CA certificate chain. "PrivateData" Property holds the Device private key paired to the certificate. (See 13.3 for additional detail regarding the "/oic/sec/cred" Resource).

9.4.5 Certificate provisioning

The CMS (e.g. a hub or a smart phone) issues certificates for new Devices.

The CA in the CMS retrieves a Device’s public key and proof of possession of the private key, generates a Device’s certificate signed by this CA certificate, and then the CMS transfers them to the Device including its CA certificate chain. Optionally, the CMS can also transfer one or more role certificates, which shall have the format described in clause 9.4.2. The "subjectPublicKey" of each role certificate shall match the "subjectPublicKey" in the Device certificate.

In the sequence in Figure 23, the Certificate Signing Request (CSR) is defined by PKCS#10 in IETF RFC 2986, and is included here by reference.

The sequence flow of a certificate transfer for a Client-directed model is described in Figure 23.

1) The CMS retrieves a CSR from the Device that requests a certificate. In this CSR, the Device shall place its requested UUID into the subject and its public key in the "SubjectPublicKeyInfo". The Device determines the public key to present; this may be an already-provisioned key it has selected for use with authentication, or if none is present, it may generate a new key pair internally and provide the public part. The key pair shall be compatible with the allowed cipher suites listed in 9.4.2.4 and 11.3.4, since the certificate will be restricted for use in OCF authentication.

2) 2) Alternatively, the CMS generates and provisions a private key and corresponding certificate directly to the Device.

3) The CMS transfers the issued certificate and CA chain to the designated Device using the same credid, to maintain the association with the private key. The credential type ("oic.sec.cred") used to transfer certificates in Figure 23 is also used to transfer role certificates, by including multiple credentials in the POST from CMS to Device. Identity certificates shall be stored with the credusage Property set to "oic.sec.cred.cert" and role certificates shall be stored with the credusage Property set to "oic.sec.cred.rolecert".

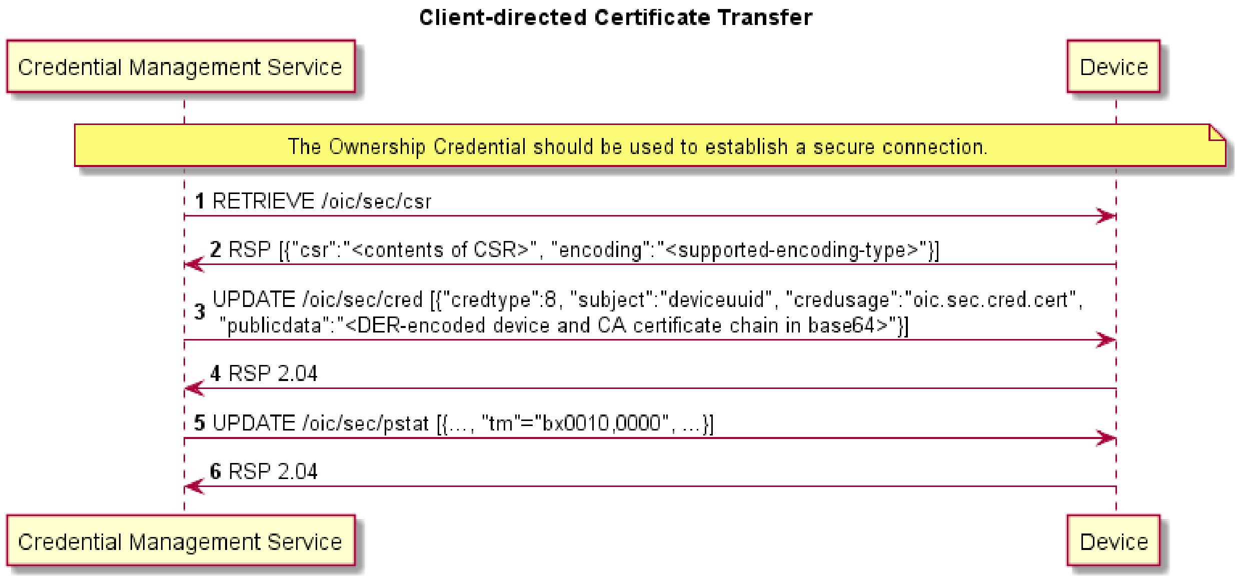

Client-directed Certificate Transfer
Credential Management Service
Device
The Ownership Credential should be used to establish a secure connection.
1 RETRIEVE /oic/sec/csr
2 RSP [{"csr":"<contents of CSR>", "encoding":"<supported-encoding-type>"}]
3 UPDATE /oic/sec/cred [{"credtype":8, "subject":"deviceuuid", "credusage":"oic.sec.cred.cert", "publicdata":"<DER-encoded device and CA certificate chain in base64>"}]
4 RSP 2.04
5 UPDATE /oic/sec/pstat [{..., "tm"="bx0010,0000", ...}]
6 RSP 2.04
Credential Management Service
Device

Figure 23 – Client-directed Certificate Transfer

9.4.6 CRL provisioning [Deprecated]

This clause is intentionally left blank.

9.4.7 Role and identity certificate profile

During onboarding, identity and optional role certificate is generated by the OBT and distributed to the Device. Table 14 is the list of required and optional fields of the certificate. If optional fields are used (from Table 14) then the device might refuse the certificate due to its size and the OBT will create a certificate that will not use the optional fields.

Table 14 - X.509 v3 extensions for role and identity certificates

Extension	Required/Optional	Criticality	Value / Remarks
keyUsage	REQUIRED	Critical	digitalSignature (0) and keyAgreement(4) bits SHALL be the only bits enabled
certificatePolicies	OPTIONAL	Non-critical	End-Entity certificates chaining to an OCF Root CA SHOULD contain at least one PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2) corresponding to the version of the OCF Certificate Policy under which it was issued. Additional manufacturer-specific CP OIDs may also be populated
extendedKeyUsage	REQUIRED	Non-critical	The following extendedKeyUsage (EKU) OIDs SHALL both be present: • serverAuthentication - 1.3.6.1.5.5.7.3.1 • clientAuthentication - 1.3.6.1.5.5.7.3.2 Exactly ONE of the following OIDs SHALL be present: • Identity certificate - 1.3.6.1.4.1.44924.1.6 • Role certificate - 1.3.6.1.4.1.44924.1.7 End-Entity certificates SHALL NOT contain the anyExtendedKeyUsage OID (2.5.29.37.0)
subjectAlternativeName	REQUIRED	Non-critical	The subjectAltName extension is used to encode one or more Role ID values in role certificates, binding the roles to the subject public key. When the extendedKeyUsage (EKU) extension contains the Identity Certificate OID (1.3.6.1.4.1.44924.1.6), the subjectAltName

			extension SHOULD NOT be present. If the EKU extension contains the Role Certificate OID (1.3.6.1.4.1.44924.1.7), the subjectAltName extension SHALL be present and populated as follows: Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name shall contain exactly one CN (Common Name) component, and zero or one OU (Organizational Unit) components. The OU component, if present, shall specify the authority that defined the semantics of the role. If the OU component is absent, the certificate issuer has defined the role. The CN component shall encode the role ID. Other GeneralName types in the SEQUENCE may be present, but shall not be interpreted as roles. The role, and authority shall be encoded as ASN.1 PrintableString type, the restricted character set [0-9a-z-A-z '()+,-./:=?].

10 Device authentication

10.1 Device authentication general

When a Client is accessing a restricted Resource on a Server, the Server shall authenticate the Client. Clients shall authenticate Servers while requesting access. Clients may also assert one or more roles that the server can use in access control decisions. Roles may be asserted when the Device authentication is done with certificates.

10.2 Device authentication with symmetric key credentials

When using symmetric keys to authenticate, the Server Device shall include the "ServerKeyExchange" message and set "psk_identity_hint" to the Server’s Device UUID. The Client shall validate that it has a credential with the Subject UUID set to the Server’s Device UUID, and a credential type of PSK. If it does not, the Client shall respond with an unknown_psk_identity error or other suitable error.

If the Client finds a suitable PSK credential, it shall reply with a "ClientKeyExchange" message that includes a "psk_identity" set to the Client’s Device UUID. The Server shall verify that it has a credential with the matching Subject UUID and type. If it does not, the Server shall respond with an "unknown_psk_identity" or other suitable error code. If it does, then it shall continue with the DTLS protocol, and both Client and Server shall compute the resulting premaster secret.

10.3 Device authentication with raw asymmetric key credentials

When using raw asymmetric keys to authenticate, the Client and the Server shall include a suitable public key from a credential that is bound to their Device. Each Device shall verify that the provided public key matches the Public Data field of a credential they have, and use the corresponding Subject UUID of the credential to identify the peer Device.

10.4 Device authentication with certificates

10.4.1 Device authentication with certificates general

When using certificates to authenticate, the Client and Server shall each include their certificate chain, as stored in the appropriate credential, as part of the selected authentication cipher suite. Each Device shall validate the certificate chain presented by the peer Device. Each certificate signature shall be verified until a public key is found within the "/oic/sec/cred" Resource with the "oic.sec.cred.trustca" credusage.

Devices shall follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In addition:

– For both End-Entity certificates and non-End-Entity certificates, Devices shall verify that "notBefore" and "notAfter" fields in the certificates conform to IETF RFC 5280 clauses 4.1.2.5, 4.1.2.5.1, and 4.1.2.5.2.

– For non-End-Entity certificates, Devices shall verify that the Basic Constraints extension is present, and that the "cA" boolean in the extension is TRUE. If any of these are false, the certificate chain shall be rejected. If the pathLenConstraint field is present, Devices shall verify that the number of certificates between this certificate and the End-Entity certificate is less than or equal to "pathLenConstraint". In particular, if "pathLenConstraint" is zero, only an End-Entity certificate can be issued by this certificate. If the "pathLenConstraint" field is absent, there is no limit to the chain length.

– For End-Entity certificates, Devices shall verify that the Basic Constraints extension (if present) has a "cA" boolean value of FALSE, and does not contain a "pathLenConstraint" ASN.1 sequence.

– For non-End-Entity certificates, Devices shall verify that the Key Usage extension is present, and that the "keyCertSign" (5) bit is asserted.

- For End-Entity certificates, Devices shall verify that the Key Usage extension is present and that "digitalSignature" (0) and "keyAgreement" (4) bits are asserted.
- For End-Entity certificates, Devices shall verify that the Extended Key Usage (EKU) extension is present and suitable to the purpose for which it is being presented: Identity ("1.3.6.1.4.1.44924.1.6") or Role ("1.3.6.1.4.1.44924.1.7"). An End-Entity certificate which contains no EKU extension, or presents both identity and role OIDs is not valid and shall be rejected. Any certificate which contains the "anyExtendedKeyUsage" purpose ("2.5.29.37.0") shall be rejected, even if other valid EKUs are also present. For End-Entity certificates, Devices shall verify that the EKU extension also contains OIDs for "serverAuthentication" ("1.3.6.1.5.5.7.3.1") and "clientAuthentication" ("1.3.6.1.5.5.7.3.2") for compatibility with various TLS implementations.
- For End-Entity certificates which chain to an OCF Root CA, the Devices should verify that they contain at least one "PolicyIdentifierId" set to the OCF Certificate Policy OID – ("1.3.6.1.4.1.51414.0.1.2") corresponding to the version of the OCF Certificate Policy under which it was issued. Additional manufacturer-specific CP OIDs may also be populated.

If the Device does not recognize an extension, it shall examine the "critical" field. If the field is TRUE, the Device shall reject the certificate. If the field is FALSE, the Device shall treat the certificate as if the extension were absent and proceed accordingly. This applies to all certificates in a chain.

A Device retrieves the Subject UUID from the "Common Name" component of the "Subject Name" property of the End-Entity certificate which has the following format: "uuid: X",, where X is provisioned by the CMS to match the "deviceuuid" Property of the "/oic/sec/doxm" Resource. The Device treats all requests arriving over a connection authenticated by this End-Entity certificate as having originated from the Device with this Subject UUID. The Device shall use this Subject UUID to match against the "subjectuuid" Property of the provisioned ACL entries to perform access control checks.

10.4.2 Role assertion with certificates

This clause describes role assertion by a client to a server using a certificate role credential.

Following authentication with a certificate, an OCF Client shall assert Roles by updating the Server’s "/oic/sec/roles" Resource with all the Role certificates it possesses, unless the device manufacturer provides a vendor-specific mechanism for End User to select which roles to assert. The Role credentials shall be certificate credentials and shall include a certificate chain. The Server shall validate each certificate chain as specified in clause 10.3. Additionally, the public key in the End-Entity certificate used for Device authentication shall be identical to the public key in all Role (End-Entity) certificates. Also, the common name component of the subject name for both Role certificates and identity certificates shall include a string of format "uuid:X" where X matches the "deviceuuid" Property of the "oic.sec.doxm" Resource.

Furthermore, a Client is prohibited from adding Role certificates for other Clients. The Server shall reject Clients’ request to add Role certificates if either (1) the request was received over an un-secured connection or (2) the request was received over a secured connection but the public key in the Role certificate does not match the public key in the identity certificate, which was used to establish the secured connection.

The Roles asserted are encoded in the "subjectAltName" extension in the certificate. The "subjectAltName" field can have multiple values, allowing a single certificate to encode multiple Roles that apply to the Client. The Server shall also check that the EKU extension of the Role certificate(s) contains the value "1.3.6.1.4.1.44924.1.7" (see clause 9.4.2.2) indicating the certificate may be used to assert Roles. Figure 24 describes how a Client Device asserts Roles to a Server.

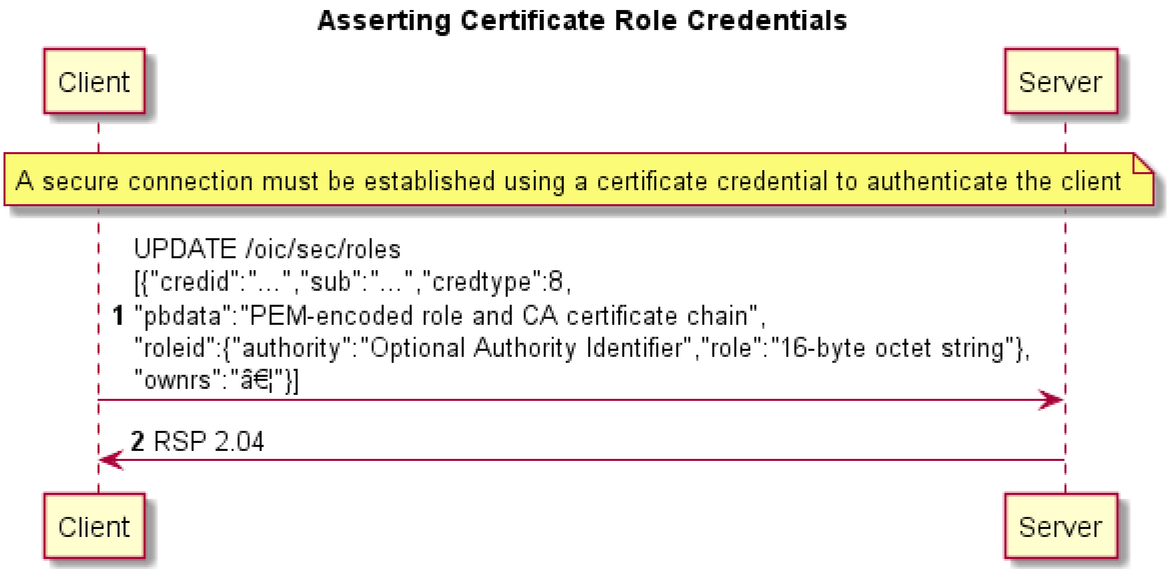

Asserting Certificate Role Credentials
Client
Server
A secure connection must be established using a certificate credential to authenticate the client
UPDATE /oic/sec/roles
[{"credid":"...","sub":"...","credtype":8,
1 "pbdata":"PEM-encoded role and CA certificate chain",
"roleid":{"authority":"Optional Authority Identifier","role":"16-byte octet string"},
"ownrs":"â€¦"}]
2 RSP 2.04
Client
Server

Figure 24 – Asserting a role with a certificate role credential.

Additional comments for Figure 24

1) The response shall contain "204 No Content" to indicate success or 4xx to indicate an error. If the server does not support certificate credentials, it should return "501 Not Implemented"

2) Roles asserted by the client may be kept for a duration chosen by the server. The duration shall not exceed the validity period of the role certificate.

3) Servers should choose a nonzero duration to avoid the cost of frequent re-assertion of a role by a client. It is recommended that servers use the validity period of the certificate as a duration, effectively allowing the CMS to decide the duration.

4) The format of the data sent in the create call shall be a list of credentials ("oic.sec.cred", see Table 20). They shall have "credtype" 8 (indicating certificates) and "PrivateData" field shall not be present. For fields that are duplicated in the "oic.sec.cred" object and the certificate, the value in the certificate shall be used for validation. For example, if the "Period" field is set in the credential, the server shall treat the validity period in the certificate as authoritative. Similar for the roleid data (authority, role).

5) Certificates shall be encoded as in Figure 23 (PEM-encoded certificate chain).

6) Clients may GET the "/oic/sec/roles" Resource to determine the roles that have been previously asserted. An array of credential objects shall be returned. If there are no valid certificates corresponding to the currently connected and authenticated Client’s identity, then an empty array (i.e. []) shall be returned.

10.4.3 OCF PKI Roots

This clause intentionally left empty.

10.4.4 PKI Trust Store

Each Device using a certificate chained to an OCF Root CA trust anchor SHALL securely store the OCF Root CA certificates in the "oic/sec/cred" Resource and SHOULD physically store this Resource in a hardened memory location where the certificates cannot be tampered with.

10.4.5 Path Validation and extension processing

See clause 10.3.

11 Message integrity and confidentiality

11.1 Preamble

Secured communications between Clients and Servers are protected against eavesdropping, tampering, or message replay, using security mechanisms that provide message confidentiality and integrity.

11.2 Session protection with DTLS

11.2.1 DTLS protection general

Devices shall support DTLS for secured communications as defined in IETF RFC 6347. Devices using TCP shall support TLS v1.2 for secured communications as defined in IETF RFC 5246. See 11.3 for a list of required and optional cipher suites for message communication.

OCF Devices shall support (D)TLS version 1.2 or greater and shall not support versions 1.1 or lower.

Multicast session semantics are not yet defined in this version of the security document.

11.2.2 Unicast session semantics

For unicast messages between a Client and a Server, both Devices shall authenticate each other. See clause 9.4.7 for details on Device Authentication.

Secured unicast messages between a Client and a Server shall employ a cipher suite from 11.3. The sending Device shall encrypt and authenticate messages as defined by the selected cipher suite and the receiving Device shall verify and decrypt the messages before processing them.

11.3 Cipher suites

11.3.1 Cipher suites general

The cipher suites allowed for use can vary depending on the context. This clause lists the cipher suites allowed during ownership transfer and normal operation. The following RFCs provide additional information about the cipher suites used in OCF.

IETF RFC 4279: Specifies use of pre-shared keys (PSK) in (D)TLS

IETF RFC 4492: Specifies use of elliptic curve cryptography in (D)TLS

IETF RFC 5489: Specifies use of cipher suites that use elliptic curve Diffie-Hellman (ECDHE) and PSKs

IETF RFC 6655 and IETF RFC 7251: Specifies AES-CCM mode cipher suites, with ECDHE

11.3.2 Cipher suites for Device Ownership Transfer

11.3.2.1 Just Works Method cipher suites

The Just Works OTM may use the following (D)TLS cipher suites.

TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256

All Devices supporting Just Works OTM shall implement:

TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256 (with the value 0xFF00)

11.3.2.2 Random PIN Method cipher suites

The Random PIN Based OTM may use the following (D)TLS cipher suites.

TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256

All Devices supporting Random Pin Based OTM shall implement:

TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256

11.3.2.3 Certificate Method cipher suites

The Manufacturer Certificate Based OTM may use the following (D)TLS cipher suites.

TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,
TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,
TLS_ECDHE_ECDSA_WITH_AES_128_CCM,
TLS_ECDHE_ECDSA_WITH_AES_256_CCM

Using the following curve:

secp256r1 (See IETF RFC 4492)

All Devices supporting Manufacturer Certificate Based OTM shall implement:

TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

Devices supporting Manufacturer Certificate Based OTM should implement:

TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,
TLS_ECDHE_ECDSA_WITH_AES_128_CCM,
TLS_ECDHE_ECDSA_WITH_AES_256_CCM

11.3.3 Cipher suites for symmetric keys

The following cipher suites are defined for (D)TLS communication using PSKs:

TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,
TLS_PSK_WITH_AES_128_CCM_8, (* 8 OCTET Authentication tag *)
TLS_PSK_WITH_AES_256_CCM_8,
TLS_PSK_WITH_AES_128_CCM, (* 16 OCTET Authentication tag *)
TLS_PSK_WITH_AES_256_CCM,

All CCM based cipher suites also use HMAC-SHA-256 for authentication.

All Devices shall implement the following:

TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

Devices should implement the following:

TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,
TLS_PSK_WITH_AES_128_CCM_8,
TLS_PSK_WITH_AES_256_CCM_8,
TLS_PSK_WITH_AES_128_CCM,
TLS_PSK_WITH_AES_256_CCM

11.3.4 Cipher suites for asymmetric credentials

The following cipher suites are defined for (D)TLS communication with asymmetric keys or certificates:

TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,

TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

TLS_ECDHE_ECDSA_WITH_AES_256_CCM

Using the following curve:

secp256r1 (See IETF RFC 4492)

All Devices supporting Asymmetric Credentials shall implement:

TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

All Devices supporting Asymmetric Credentials should implement:

TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

TLS_ECDHE_ECDSA_WITH_AES_256_CCM

12 Access control

12.1 ACL generation and management

This clause intentionally left empty.

12.2 ACL evaluation and enforcement

12.2.1 ACL evaluation and enforcement general

The Server enforces access control over application Resources before exposing them to the requestor. The Security Layer in the Server authenticates the requestor when access is received via the secure port. Authenticated requestors, known as the "subject" can be used to match ACL entries that specify the requestor's identity, role or may match authenticated requestors using a subject wildcard.

If the request arrives over the unsecured port, the only ACL policies allowed are those that use a subject wildcard match of anonymous requestors.

Access is denied if a requested Resource is not matched by an ACL entry.

NOTE There are documented exceptions pertaining to Device onboarding where access to Security Virtual Resources may be granted prior to provisioning of ACL Resources.

The second generation ACL (i.e. "/oic/sec/acl2") contains an array of Access Control Entries (ACE2) that employ a Resource matching algorithm that uses an array of Resource references to match Resources to which the ACE2 access policy applies. Matching consists of comparing the values of the ACE2 "resources" Property (see clause 13) to the requested Resource. Resources are matched in two ways:

1) host reference ("href")
2) Resource wildcard ("wc").

12.2.2 Host reference matching

When present in an ACE2 matching element, the Host Reference (href) Property shall be used for Resource matching.

– The href Property shall be used to find an exact match of the Resource name if present.

12.2.3 Resource wildcard matching

When present, a wildcard ("wc") expression shall be used to match multiple Resources using a wildcard Property contained in the "oic.sec.ace2.resource-ref" structure.

A wildcard expression may be used to match multiple Resources using a wildcard Property contained in the "oic.sec.ace2.resource-ref" structure. The wildcard matching strings are defined in Table 15.

Table 15 – ACE2 wildcard matching strings description

String	Description
"+"	Shall match all Discoverable Non-Configuration Resources which expose at least one Secure OCF Endpoint.
"-"	Shall match all Discoverable Non-Configuration Resources which expose at least one Unsecure OCF Endpoint.
"*"	Shall match all Non-Configuration Resources.

NOTE Discoverable Resources appear in the "/oic/res" Resource, while non-discoverable Resources may appear in other collection Resources but do not appear in the /res collection.

12.2.4 Multiple criteria matching

If the ACE2 "resources" Property contains multiple entries, then a logical OR shall be applied for each array element. For example, if a first array element of the "resources" Property contains "href"="/a/light" and the second array element of the "resources" Property contains "href"="/a/led", then Resources that match either of the two "href" criteria shall be included in the set of matched Resources.

Example 1 JSON for Resource matching

```
{
//Matches Resources named "/x/door1" or "/x/door2"
  "resources":[
     {
        "href":"/x/door1"
     },
     {
        "href":"/x/door2"
     },
  ]
}
```

Example 2 JSON for Resource matching

```
{
  // Matches all Resources
    "resources":[
     {
            "wc":"*"
     }
  ]
}
```

12.2.5 Subject matching using wildcards

When the ACE subject is specified as the wildcard string "*" any requestor is matched. The OCF server may authenticate the OCF client, but is not required to.

Examples: JSON for subject wildcard matching

```
//matches all subjects that have authenticated and confidentiality protections in place.
"subject" : {
   "conntype" : "auth-crypt"
}
//matches all subjects that have NOT authenticated and have NO confidentiality protections in place.
"subject" : {
    "conntype" : "anon-clear"
}
```

12.2.6 Subject matching using roles

When the ACE subject is specified as a role, a requestor shall be matched if either:

1) The requestor authenticated with a symmetric key credential, and the role is present in the "roleid" Property of the credential’s entry in the "credential" Resource, or

2) The requestor authenticated with a certificate, and a valid role certificate is present in the roles Resource with the requestor's certificate's public key at the time of evaluation. Validating role certificates is defined in 10.3.1.

12.2.7 ACL evaluation

12.2.7.1 ACE2 matching algorithm

The OCF Server shall apply an ACE2 matching algorithm that matches in the following sequence:

1) The local "/oic/sec/acl2" Resource contributes its ACE2 entries for matching.
2) Access shall be granted when all these criteria are met:
   a) The requestor is matched by the ACE2 "subject" Property.
   b) The requested Resource is matched by the ACE2 "resources" Property and the requested Resource shall exist on the local Server.
   c) The "period" Property constraint shall be satisfied.
   d) The "permission" Property constraint shall be applied.

If multiple ACE2 entries match the Resource request, the union of permissions, for all matching ACEs, defines the effective permission granted. E.g. If Perm1=CR---; Perm2=--UDN; Then UNION (Perm1, Perm2)=CRUDN.

The Server shall enforce access based on the effective permissions granted.

Batch requests to Resource containing Links require additional considerations when accessing the linked Resources. ACL considerations for batch request to the Atomic Measurement Resource Type are provided in clause 12.2.7.2. ACL considerations for batch request to the Collection Resource Type are provided in clause 12.2.7.3.

Clause 12.2.7.4 provides ACL considerations when a new Resource is created on a Server in response to a CREATE request.

12.2.7.2 ACL considerations for batch request to the Atomic Measurement Resource Type

The present clause shall apply to any Resource Type based on the Atomic Measurement Resource Type.

If an OCF Server receives a batch OCF Interface request to an Atomic Measurement Resource and there is an ACE matching the Atomic Measurement Resource which permits the request, then the corresponding requests to the linked Resources of the Atomic Measurement Resource shall be permitted by the OCF Server. That is, the request to each linked Resource is permitted regardless of whether there is an ACE configured on the OCF Server which would permit a corresponding request from the OCF Client (which sent the batch OCF Interface request to the Atomic Measurement Resource) addressing the linked Resource.

NOTE As specified in ISO/IEC 30118-1, the linked Resources of an Atomic Measurement Resource are hosted on the same Device as the Atomic Measurement Resource.

12.2.7.3 ACL considerations for a batch OCF Interface request to a Collection

This cluase addresses the additional authorization processes which take place when a Server receives a batch OCF Interface request from a Client to a Collection hosted on that Server, assuming there is an ACE matching the Collection which permits the original Client request. For the purposes of this cluase, the Server hosting this Collection is called the "Collection host". The additional authorization process is dependent on whether the linked Resource is hosted on the Collection host or the linked Resource is hosted on another Server:

- For each generated request to a linked Resource hosted on the Collection host, the Collection host shall apply the ACE2 matching algorithm in clause 12.2.7.1 to determine whether the linked Resource is permitted to process the generated request, with the following clarifications:
  - The requestor in clause 12.2.7.1 shall be the Client which sent the original Client request.
  - The requested Resource in clause 12.2.7.1 shall be the linked Resource, which shall be matched using at least one of:
    - a Resource Wildcard matching the linked Resource, or
    - an exact match of the local path of the linked Resource with a "href" Property in the "resources" array in the ACE2.
    - an exact match of the full URI of the linked Resource with a "href" Property in the "resources" array in the ACE2.

NOTE The full URI of a linked Resource is obtained by concatenating the "anchor" Property of the Link, if present, and the "href" Property of the Link. The local path can then be determined form the full URI.

If the linked Resource is not permitted to process the generated request, then the Collection host shall treat such cases as a linked Resource which cannot process the request when composing the aggregated response to the original Client Request, as specified for the batch OCF Interface in the ISO/IEC 30118-1.

12.2.7.4 ACL considerations on creation of a new Resource

When a new Resource is created on a Server in response to a CREATE request, there might be no ACEs permitting access to the newly created Resource. The present clause describes how the Server autonomously modifies the "/oic/sec/acl2" Resource to provide some initial authorizations for accessing the newly created Resource. The purpose of this autonomous modification is to avoid relying on the AMS update the "/oic/sec/acl2" Resource after every new Resource is created.

Subsequent to a Server creating a Collection inside another Collection in response to a CREATE request from a Client, and prior to sending a response to the Client:

- If there is an ACE with "subject" containing the UUID of the Client, and "permissions" exactly matching the CREATE, RETRIEVE, UPDATE and DELETE operations, then the Server shall autonomously add an "href" entry to "resources" with the URI of the newly created Collection.
- Otherwise, the Server shall autonomously add an ACE with "subject" containing the UUID of the Client, "resources" containing an "href" entry with the URI of the newly created Collection, and "permissions" exactly matching the CREATE, RETRIEVE, UPDATE and DELETE operations.

Subsequent to a Server creating a non-Collection Resource inside another Collection in response to a CREATE request from a Client, and prior to sending a response to the Client:

- If there is an ACE with "subject" containing the UUID of the Client, and "permissions" exactly matching the RETRIEVE, UPDATE and DELETE operations, then the Server shall autonomously add an "href" entry to "resources" with the URI of the newly created Resource.
- Otherwise, the Server shall autonomously add an ACE with "subject" containing the UUID of the Client, "resources" containing an "href" entry with the URI of the newly created, and "permissions" exactly matching the RETRIEVE, UPDATE and DELETE operations.

13 Security Resources

13.1 Security Resources general

OCF Security Resources are shown in Figure 25.

"/oic/sec/cred" Resource and Properties are shown in Figure 26.

"/oic/sec/acl2" Resource and Properties are shown in Figure 27.

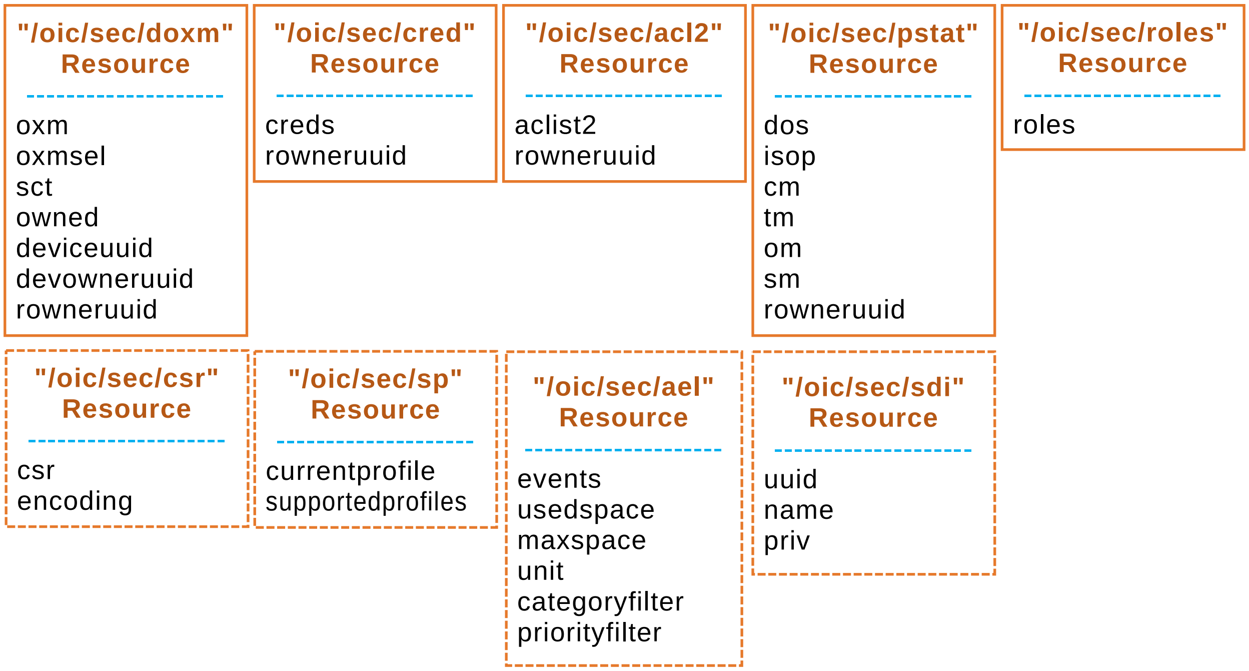
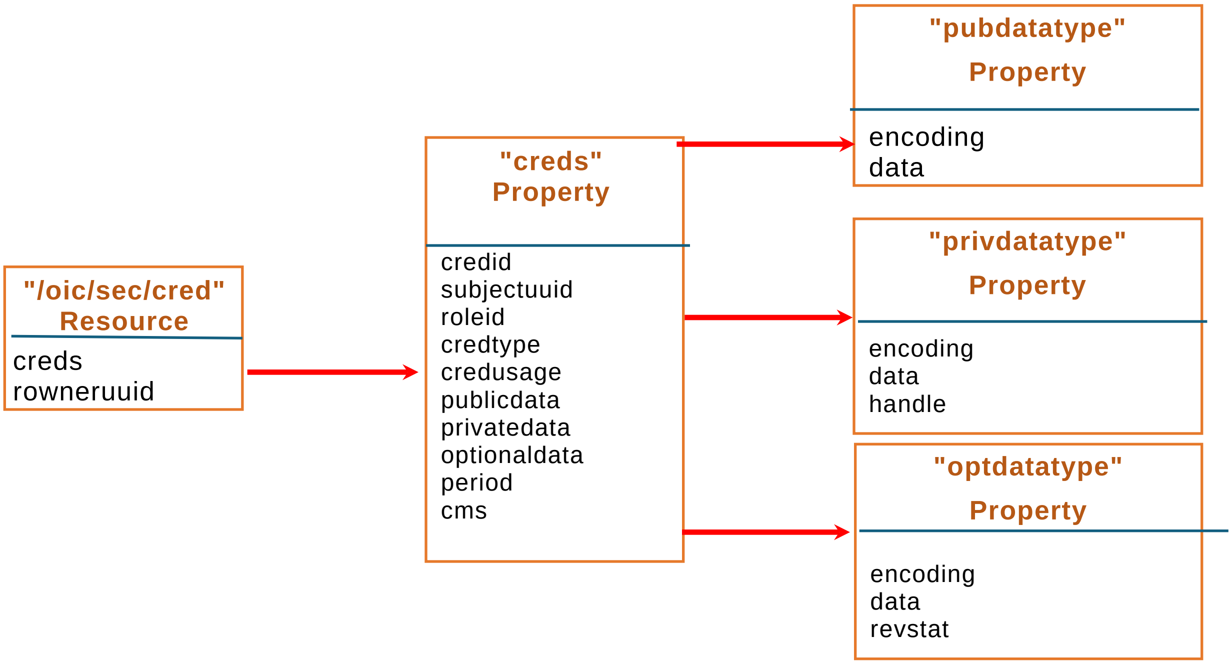
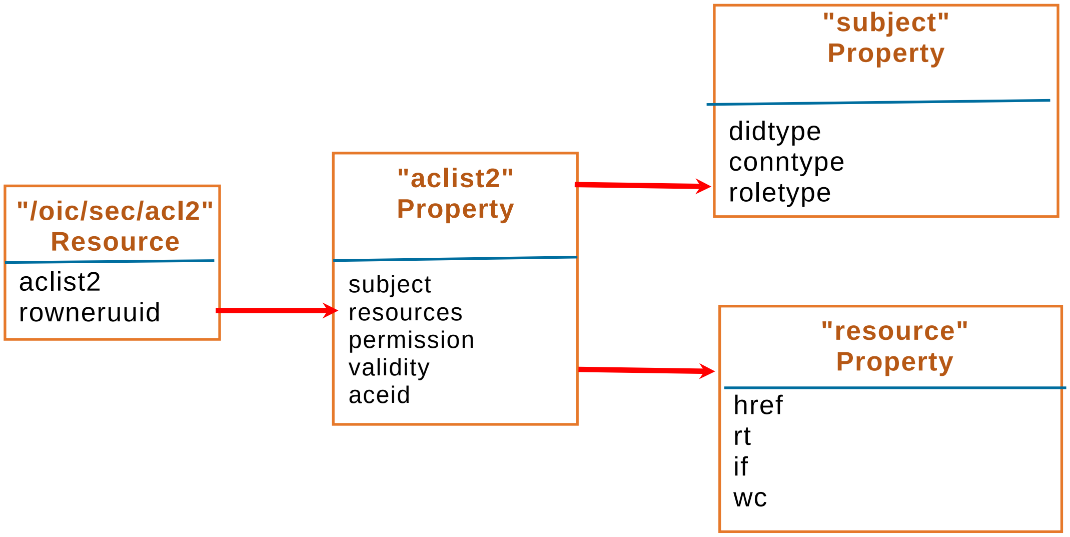


Figure 25 – OCF Security Resources



2821 **Figure 26 – "/oic/sec/cred" Resource and Properties**

2822



2823 **Figure 27 – "/oic/sec/acl2" Resource and Properties**

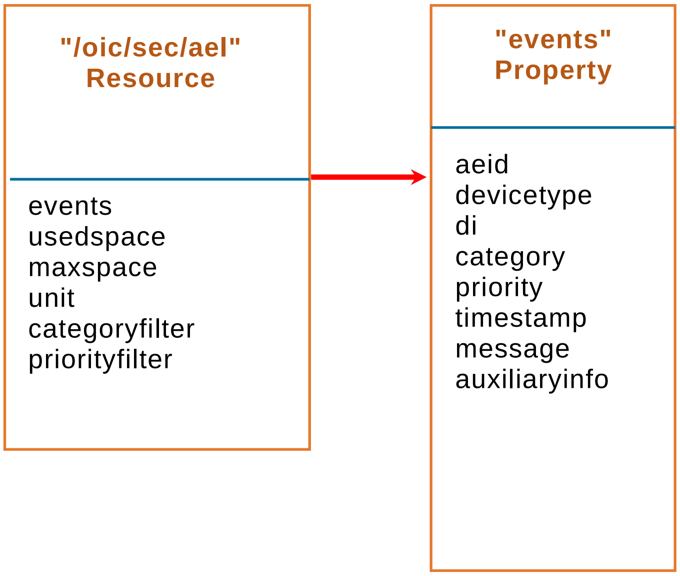


Figure 28 – "/oic/sec/ael" Resource and Properties

13.2 Device Owner Transfer Resource

13.2.1 Device Owner Transfer Resource general

The "/oic/sec/doxm" Resource contains the set of supported Device OTMs.

Resource discovery processing respects the CRUDN constraints supplied as part of the security Resource definitions contained in this document.

"/oic/sec/doxm" Resource is defined in Table 16.

Table 16 – Definition of the "/oic/sec/doxm" Resource

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/doxm	Device OTMs	oic.r.doxm	oic.if.baseline, oic.if.rw	Resource for supporting Device owner transfer	Configuration

Table 17 defines the Properties of the "/oic/sec/doxm" Resource.

Table 17 – Properties of the "/oic/sec/doxm" Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
OTM	oxms	oic.sec.doxmtype	array	Yes		R	Value identifying the owner-transfer-method and the organization that defined the method.
OTM Selection	oxmsel	oic.sec.doxmtype	UINT16	Yes	RESET	R	Server shall set to (4) "oic.sec.oxm.self"
					RFOTM (no open DOC)	RW	DOTS shall set to its selected DOTS and both parties execute the DOTS. After secure owner transfer session is established DOTS shall update the oxmsel again making it permanent. If the

							DOTS fails the Server shall transition device state to RESET.
					RFOTM (open DOC)	R	n/a
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	R	n/a
Supported Credential Types	sct	oic.sec.credtype	bitmask	Yes		R	Identifies the types of credentials the Device supports. The Server sets this value at framework initialization after determining security capabilities. The Device always supports symmetric pair-wise key and asymmetric signing key with certificate (bit positions 0x1 and 0x8 respectively). Other credential types are optional as per clause 9.3
Device Ownership Status	owned	Boolean	T|F	Yes	RESET	R	Server shall set to FALSE.
					RFOTM (no open DOC)	R	FALSE
					RFOTM (open DOC)	RW	DOTS (Device communicating over DOC) shall set to TRUE after secure owner transfer session is established.
					RFPRO	R	TRUE
					RFNOP	R	TRUE
					SRESET	R	TRUE
Device UUID	deviceuuid	String	oic.sec.didtype	Yes	RESET	R	No stipulation.
					RFOTM (no open DOC)	R	n/a
					RFOTM (open DOC)	RW	DOTS (Device communicating over DOC) updates to a value it has selected after secure owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	R	n/a
Device Owner Id	devowneruuid	String	uuid	Yes	RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
					RFOTM (no open DOC)	R	n/a
					RFOTM (open DOC)	RW	DOTS (Device communicating over DOC) shall set value after secure owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	R	n/a

Resource Owner Id	rowneruuid	String	uuid	Yes	RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
					RFOTM (no open DOC)	R	n/a
					RFOTM (open DOC)	RW	The DOTS (Device communicating over DOC) shall configure the rowneruuid Property when a successful owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	RW	The DOTS (referenced via devowneruuid Property) should verify and if needed, update the Resource owner Property when a mutually authenticated secure session is established. If the rowneruuid does not refer to a valid DOTS device identifier the Server shall transition to RESET.

Table 18 defines the Properties of the "oic.sec.didtype".

Table 18 – Properties of the "oic.sec.didtype" type

Property Title	Property Name	Value Type	Value Rule	Mand atory	Device State	Access Mode	Description
Device UUID	uuid	String	uuid	Yes	RW	-	A uuid value

The "oxms" Property contains a list of OTM where the entries appear in the order of preference. This Property contains the higher priority methods appearing before the lower priority methods. The DOTS queries this list at the time of onboarding and selects the most appropriate method.

OTMs consist of two parts, a URI identifying the vendor or organization and the specific method.

```
<DoxmType> ::= <NSS>
<NSS> ::= <Identifier> | {{<NID>"."} <NameSpaceQualifier> "."} <Method>
<NID> :: = <Vendor-or-Organization>
<Identifier> ::= INTEGER
<NameSpaceQualifier> ::= String
<Method> ::= String
<Vendor-Organization> ::= String
```

When an OTM successfully completes, the "owned" Property is set to "1" (TRUE). Consequently, subsequent attempts to take ownership of the Device will fail.

There are four device identifiers:

1) "deviceuuid" Property of "/oic/sec/doxm" Resource - random DOTS-provisioned value unique for a given security domain, used as a device identity for access control, mapped internally to a device-owned credential.
2) "di" Property of "/oic/d" Resource - mirroring the value of "deviceuuid" Property of "/oic/sec/doxm" Resource.
3) "piid" Property of "/oic/d" Resource - defined in ISO/IEC 30118-1.
4) "pi" Property of "/oic/p" Resource - defined in ISO/IEC 30118-1.

The "/oic/sec/doxm" Resource supports CoAP multicast requests in certain cases. For details see clause 7.3.1

13.2.2 OCF defined OTMs

Table 19 defines the Properties of the "oic.sec.doxmtype".

Table 19 – Properties of the "oic.sec.doxmtype" type

Value Type Name	Value Type URN (optional)	Enumeration Value (mandatory)	Description
OCFJustWorks	oic.sec.doxm.jw	0	The just-works method relies on anonymous Diffie-Hellman key agreement protocol to allow a DOTS to assert ownership of the new Device. The first DOTS to make the assertion is accepted as the Device owner. The just-works method results in a shared secret that is used to authenticate the Device to the DOTS and likewise authenticates the DOTS to the Device. The Device permits the DOTS to take ownership of the Device, after which a second attempt to take ownership by a different DOTS will fail[a].
OCFSharedPin	oic.sec.doxm.rdp	1	The new Device randomly generates a PIN that is communicated via an Out Of Band Communication Channel to a DOTS. An in-band Diffie-Hellman key agreement protocol establishes that both endpoints possess the PIN. Possession of the PIN by the DOTS signals the new Device that device ownership can be asserted.
OCFMfgCert	oic.sec. doxm.mfgcert	2	The new Device is presumed to have been manufactured with an embedded asymmetric private key that is used to sign a Diffie-Hellman exchange at Device onboarding. The manufacturer certificate should contain Platform hardening information and other security assurances assertions.
OCF Reserved	<Reserved>	3	Reserved
OCFSelf	oic.sec.oxm.self	4	The manufacturer shall set the "/doxm.oxmsel" value to (4). The Server shall reset this value to (4) upon entering RESET.
OCF Reserved	<Reserved>	5~0xFEFF	Reserved for OCF use
Vendor-defined Value Type Name	<Reserved>	0xFF00~0xFFFF	Reserved for vendor-specific OTM use

[a] The just-works method is subject to a man-in-the-middle attacker. Precautions should be taken to provide physical security when this method is used.

13.3 Credential Resource

13.3.1 Credential Resource general

The "/oic/sec/cred" Resource maintains credentials used to authenticate the Server to Clients and support services as well as credentials used to verify Clients and support services.

Multiple credential types are anticipated by the OCF framework, including pair-wise pre-shared keys, asymmetric keys, certificates and others. The credential Resource uses a Subject UUID to distinguish the Clients and support services it recognizes by verifying an authentication challenge.

In order to provide an interface which allows management of the "creds" Array Property, the RETRIEVE, UPDATE and DELETE operations on the "/oic/sec/cred" Resource shall behave as follows:

1) A RETRIEVE shall return the full Resource representation, except that any write-only Properties shall be omitted (e.g. private key data).

2) An UPDATE shall replace or add to the Properties included in the representation sent with the UPDATE request, as follows:

   a) If an UPDATE representation includes the "creds" array Property, then:

      i) Supplied "creds" with a "credid" that matches an existing "credid" shall replace completely the corresponding "cred" in the existing "creds" array.

      ii) Supplied "creds" without a "credid" shall be appended to the existing "creds" array, and a unique (to the "cred" Resource) "credid" shall be created and assigned to the new "cred" by the Server. The "credid" of a deleted "cred" should not be reused, to improve the determinism of the interface and reduce opportunity for race conditions.

      iii) Supplied "creds" with a "credid" that does not match an existing "credid" shall be appended to the existing "creds" array, using the supplied "credid".

      iv) The rows in Table 21 corresponding to the "creds" array Property dictate the Device States in which an UPDATE of the "creds" array Property is always rejected. If OCF Device is in a Device State where the Access Mode in this row contains "R", then the OCF Device shall reject all UPDATEs of the "creds" array Property.

3) A DELETE without query parameters shall set the "creds" array to the empty array, but shall not remove the "/oic/sec/cred" Resource.

4) A DELETE with one or more "credid" query parameters shall remove the "cred"(s) with the corresponding "credid"(s) from the "creds" array.

5) The rows in Table 21 corresponding to the "creds" array Property dictate the Device States in which a DELETE is always rejected. If OCF Device is in a Device State where the Access Mode in this row contains "R", then the OCF Device shall reject all DELETEs.

NOTE The "/oic/sec/cred" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined in ISO/IEC 30118-1.

"/oic/sec/cred" Resource is defined in Table 20.

Table 20 – Definition of the "/oic /sec/cred" Resource

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/cred	Credentials	oic.r.cred	oic.if.baseline, oic.if.rw	Resource containing credentials for Device authentication, verification and data protection	Security

Table 21 defines the Properties of the "/oic/sec/cred" Resource.

Table 21 – Properties of the "/oic/sec/cred" Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
Credentials	creds	oic.sec.cred	array	Yes	RESET	R	Server shall set to manufacturer defaults.
					RFOTM	RW	Set by DOTS after successful OTM
					RFPRO	RW	Set by the CMS (referenced via the rowneruuid Property of "/oic/sec/cred" Resource) after successful authentication. Access to NCRs is prohibited.
					RFNOP	R	Access to NCRs is permitted after a matching ACE is found.
					SRESET	RW	The DOTS (referenced via devowneruuid Property of "/oic/sec/doxm" Resource or the rowneruuid Property of "/oic/sec/doxm" Resource) should evaluate the integrity of and may update creds entries when a secure session is established and the Server and DOTS are authenticated.
Resource Owner ID	rowneruuid	String	uuid	Yes	RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
					RFOTM	RW	The DOTS shall configure the rowneruuid Property of "/oic/sec/cred" Resource when a successful owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	RW	The DOTS (referenced via devowneruuid Property of "/oic/sec/doxm" Resource or the rowneruuid Property of "/oic/sec/doxm" Resource) should verify and if needed, update the Resource owner Property when a mutually authenticated secure session is established. If the "rowneruuid" Property does not refer to a valid DOTS the Server shall transition to RESET.

All secure Device accesses shall have a "/oic/sec/cred" Resource that protects the end-to-end interaction.

The "/oic/sec/cred" Resource shall be updateable by the service named in its rowneruuid Property.

ACLs naming "/oic/sec/cred" Resource should further restrict access beyond CRUDN access modes.

Table 22 defines the Properties of "oic.sec.creds".

Table 22 – Properties of the "oic.sec.creds" Property

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Credential ID	credid	UINT16	0 – 64K-1	Yes	RW		Short credential ID for local references from other Resource
Subject UUID	subjectuuid	String	uuid	Yes	RW		A uuid that identifies the subject to which this credential applies or "*" if any identity is acceptable
Role ID	roleid	oic.sec.roletype	-	No	RW		Identifies the role(s) the subject is authorized to assert.
Credential Type	credtype	oic.sec.credtype	bitmask	Yes	RW		Represents this credential's type. 0 – Used for testing 1 – Symmetric pair-wise key 2 – Symmetric group key 4 – Asymmetric signing key 8 – Asymmetric signing key with certificate 16 – PIN or password 32 – Asymmetric encryption key 64 – Directly Provisioned OSCORE Security Context 128 – Simple Secure Multicast Client Context 256 – Simple Secure Multicast Server Context
Credential Usage	credusage	oic.sec.credusagetype	String	No	RW		Used to resolve undecidability of the credential. Provides indication for how/where the cred is used "oic.sec.cred.trustca": certificate trust anchor "oic.sec.cred.cert": identity certificate "oic.sec.cred.rolecert": role certificate "oic.sec.cred.mfgtrustca": manufacturer certificate trust anchor "oic.sec.cred.mfgcert": manufacturer certificate
Public Data	publicdata	oic.sec.pubdatatype	-	No	RW		Credential Type dependent. Public credential information 1:2: ticket, public SKDC values 4, 32: Public key value 8: A chain of one or more certificate
Private Data	privatedata	oic.sec.privdatatype	-	No	-	RESET	Server shall set to manufacturer default
					RW	RFOTM	Set by DOTS after successful OTM
					W	RFPRO	Set by authenticated DOTS or CMS
					-	RFNOP	Not writable during normal operation.
					W	SRESET	DOTS may modify to enable transition to RFPRO.
Optional Data	optionaldata	oic.sec.optdatatype	-	No	RW		Credential Type dependent. Credential revocation status information 1, 2, 4, 32, 64: revocation status information 8: Revocation information

Period	period	String	-	No	RW		Period as defined by IETF RFC 5545. The credential should not be used if the current time is outside the Period window.
Credential Refresh Method	crms	oic.sec.crmtype	array	No	RW		Credentials with a Period Property are refreshed using the credential refresh method (crm) according to the type definitions for "oic.sec.crm".
OSCORE Configuration	oscore	oic.sec.oscoretype		No	RW		Contains parameters for use with credentials intended for use with OSCORE. See type definition for "oic.sec.oscoretype"

Table 23 defines the Properties of "oic.sec.credusagetype".

Table 23: Properties of the "oic.sec.credusagetype" Property

Value Type Name	Value Type URN (mandatory)
Trust Anchor	oic.sec.cred.trustca
Certificate	oic.sec.cred.cert
Role Certificate	oic.sec.cred.rolecert
Manufacturer Trust CA	oic.sec.cred.mfgtrustca
Manufacturer CA	oic.sec.cred.mfgcert

Table 24 defines the Properties of "oic.sec.pubdatatype".

Table 24 – Properties of the "oic.sec.pubdatatype" Property

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Encoding format	encoding	String	N/A	RW	No	A string specifying the encoding format of the data contained in the pubdata "oic.sec.encoding.pem" – Encoding for PEM-encoded certificate or chain
Data	data	String	N/A	RW	No	The encoded value

Table 25 defines the Properties of "oic.sec.privdatatype".

Table 25 – Properties of the "oic.sec.privdatatype" Property

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Encoding format	encoding	String	N/A	RW	Yes	A string specifying the encoding format of the data contained in the privdata "oic.sec.encoding.pem" – Encoding for PEM-encoded private key "oic.sec.encoding.base64" – Encoding of Base64 encoded PSK "oic.sec.encoding.handle" – Data is contained in a storage sub-system referenced using a handle "oic.sec.encoding.raw" – Raw hex encoded data
Data	data	String	N/A	W	No	The encoded value This value shall not be RETRIEVE-able.
Handle	handle	UINT16	N/A	RW	No	Handle to a key storage Resource

Table 26 defines the Properties of "oic.sec.optdatatype".

Table 26 – Properties of the "oic.sec.optdatatype" Property

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Revocation status	revstat	Boolean	T | F	RW	Yes	Revocation status flag True – revoked False – not revoked
Encoding format	encoding	String	N/A	RW	No	A string specifying the encoding format of the data contained in the optdata "oic.sec.encoding.pem" – Encoding for PEM-encoded certificate or chain
Data	data	String	N/A	RW	No	The encoded structure

Table 27 defines the Properties of "oic.sec.roletype".

Table 27 – Definition of the "oic.sec.roletype" type.

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Authority	authority	String	N/A	R	No	A name for the authority that defined the role. If not present, the credential issuer defined the role. If present, is expressible as an ASN.1 PrintableString.
Role	role	String	N/A -	R	Yes	An identifier for the role. Is expressible as an ASN.1 PrintableString.

Table 28 defines the Properties of "oic.sec.oscoretype".

Table 28 – Definition of the "oic.sec.oscoretype" type.

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
OSCORE Sender ID	senderid	String	Hexadecimal encoding	RW	No	OSCORE Sender ID for this OSCORE Security Context.
OSCORE Recipient ID	recipientid	String		RW	No	OSCORE Recipient ID for this OSCORE Security Context.
OSCORE Sender Sequence Number 1	ssn	Integer		R	No	OSCORE Sender Sequence Number being stored in non volatile memory to handle the loss of mutable security context parameters. See clause 16.2.4.
OSCORE Security Context Description	desc	String		RW	No	Description of the usage of this OSCOE Security Context.

13.3.2 Properties of the Credential Resource

13.3.2.1 Credential ID

Credential ID ("credid") is a local reference to an entry in a "creds" Property array of the "/oic/sec/cred" Resource. The SRM generates it. The "credid" Property shall be used to disambiguate array elements of the "creds" Property.

13.3.2.2 Subject UUID

The "subjectuuid" Property identifies the Device to which an entry in a "creds" Property array of the "/oic/sec/cred" Resource shall be used to establish a secure session, verify an authentication challenge-response or to authenticate an authentication challenge.

A "subjectuuid" Property that matches the Server’s own "deviceuuid" Property, distinguishes the array entries in the "creds" Property that pertain to this Device.

The "subjectuuid" Property shall be used to identify a group to which a group key is used to protect shared data.

When certificate chain is used during secure connection establishment, the "subjectuuid" Property shall also be used to verify the identity of the responder. The presented certificate chain shall be accepted, if there is a matching Credential entry on the Device that satisfies all of the following:

– Public Data of the entry contains trust anchor (root) of the presented chain.

– Subject UUID of the entry matches UUID in the Common Name field of the End-Entity certificate in the presented chain. If Subject UUID of the entry is set as a wildcard "*", this condition is automatically satisfied.

– Credential Usage of the entry is "oic.sec.cred.trustca".

13.3.2.3 Role ID

The "roleid" Property identifies a role that has been granted to the credential.

13.3.2.4 Credential type

The "credtype" Property is used to interpret several of the other Property values whose contents can differ depending on credential type. These Properties include "publicdata", "privatedata" and "optionaldata". The "credtype" Property value of "0" ("no security mode") is reserved for testing and debugging circumstances. Production deployments shall not allow provisioning of credentials of type "0". The SRM should introduce checking code that prevents its use in production deployments.

13.3.2.5 Public data

The "publicdata" Property contains information that provides additional context surrounding the issuance of the credential. For example, it might contain information included in a certificate or response data from a CMS. It might contain wrapped data.

13.3.2.6 Private data

The "privatedata" Property contains secret information that is used to authenticate a Device, protect data or verify an authentication challenge-response.

The "privatedata" Property shall not be disclosed outside of the SRM’s trusted computing perimeter. A secure element (SE) or trusted execution environment (TEE) should be used to implement the SRM’s trusted computing perimeter. The privatedata contents may be referenced using a handle; for example, if used with a secure storage sub-system.

13.3.2.7 Optional data

The "optionaldata" Property contains information that is optionally supplied, but facilitates key management, scalability or performance optimization.

13.3.2.8 Period

The "period" Property identifies the validity period for the credential. If no validity period is specified, the credential lifetime is undetermined. Constrained devices that do not implement a date-time capability shall obtain current date-time information from its CMS.

13.3.2.9 Credential Refresh Method type definition [Deprecated]

This clause is intentionally left blank.

13.3.2.10 Credential usage

Credential Usage indicates to the Device the circumstances in which a credential should be used. Five values are defined:

- "oic.sec.cred.trustca": This certificate is a trust anchor for the purposes of certificate chain validation, as defined in 10.4. OCF Server SHALL remove any "/oic/sec/cred" entries with an "oic.sec.cred.trustca" credusage upon transitioning to RFOTM. OCF Servers SHALL use "/oic/sec/cred" entries that have an "oic.sec.cred.trustca" Value of “credusage” Property only as trust anchors for post-onboarding (D)TLS session establishment in RFNOP; these entries are not to be used for onboarding (D)TLS sessions.
- "oic.sec.cred.cert": This "credusage" is used for certificates for which the Device possesses the private key and uses it for identity authentication in a secure session, as defined in clause 10.4.
- "oic.sec.cred.rolecert": This "credusage" is used for certificates for which the Device possesses the private key and uses to assert one or more roles, as defined in clause 10.4.2.
- "oic.sec.cred.mfgtrustca": This certificate is a trust anchor for the purposes of the Manufacturer Certificate Based OTM as defined in clause 7.3.6. OCF Servers SHALL use "/oic/sec/cred" entries that have an "oic.sec.cred.mfgtrustca" Value of "credusage" Property only as trust anchors for onboarding (D)TLS session establishment; these entries are not to be used for post-onboarding (D)TLS sessions.
- "oic.sec.cred.mfgcert": This certificate is used for certificates for which the Device possesses the private key and uses it for authentication in the Manufacturer Certificate Based OTM as defined in clause 7.3.6.

13.3.2.11 Resource Owner

The Resource Owner Property allows credential provisioning to occur soon after Device onboarding before access to support services has been established. It identifies the entity authorized to manage the "/oic/sec/cred" Resource in response to Device recovery situations.

13.3.3 Key formatting

13.3.3.1 Symmetric key formatting

Symmetric keys shall have the format described in Table 29 and Table 30.

Table 29 – 128-bit symmetric key

Name	Value	Type	Description
Length	16	OCTET	Specifies the number of 8-bit octets following Length
Key	opaque	OCTET Array	16-byte array of octets. When used as input to a PSK function Length is omitted.

Table 30 – 256-bit symmetric key

Name	Value	Type	Description
Length	32	OCTET	Specifies the number of 8-bit octets following Length
Key	opaque	OCTET Array	32-byte array of octets. When used as input to a PSK function Length is omitted.

13.3.3.2 Asymmetric keys

Asymmetric key formatting is not available in this revision of the document.

13.3.3.3 Asymmetric keys with certificate

Key formatting is defined by certificate definition.

13.3.3.4 Passwords

Password formatting is not available in this revision of the document.

13.3.4 Credential Refresh Method details [Deprecated]

This clause is intentionally left blank.

13.4 Certificate Revocation List

13.4.1 CRL Resource definition [Deprecated]

This clause is intentionally left blank.

13.5 ACL Resources

13.5.1 ACL Resources general

All Resource hosted by a Server are required to match an ACL policy. ACL policies can be expressed using "/oic/sec/acl2". The subject (e.g. "deviceuuid" of the Client) requesting access to a Resource shall be authenticated prior to applying the ACL check. Resources that are available to multiple Clients can be matched using a wildcard subject. All Resources accessible via the unsecured communication endpoint shall be matched using a wildcard subject.

13.5.2 OCF Access Control List (ACL) BNF defines ACL structures.

ACL structure in Backus-Naur Form (BNF) notation is defined in Table 31:

Table 31 – BNF definition of OCF ACL

<ACL>	<ACE> {<ACE>}
<ACE>	<SubjectId> <ResourceRef> <Permission> {<Validity>}
<SubjectId>	<DeviceId> | <Wildcard> | <RoleId>
<DeviceId>	<UUID>
<RoleId>	<Character> | <RoleName><Character>
<RoleName>	"" | <Authority><Character>
<Authority>	<UUID>
<ResourceRef>	' (' <OIC_LINK> {',' {OIC_LINK>} ')'
<Permission>	('C' | '-') ('R' | '-') ('U' | '-') ('D' | '-') ('N' | '-')
<Validity>	<Period> {<Recurrence>}
<Wildcard>	'*'
<URI>	IETF RFC 3986
<UUID>	IETF RFC 4122
<Period>	IETF RFC 5545 Period
<Recurrence>	IETF RFC 5545 Recurrence
<OIC_LINK>	ISO/IEC 30118-1 defined in JSON Schema
<Character>	<Any UTF8 printable character, excluding NUL>

The <DeviceId> token means the requestor must possess a credential that uses <UUID> as its identity in order to match the requestor to the <ACE> policy.

The <RoleID> token means the requestor must possess a role credential with <Character> as its role in order to match the requestor to the <ACE> policy.

The <Wildcard> token "*" means any requestor is matched to the <ACE> policy, with or without authentication.

When a <SubjectId> is matched to an <ACE> policy the <ResourceRef> is used to match the <ACE> policy to Resources.

The <OIC_LINK> token contains values used to query existence of hosted Resources.

The <Permission> token specifies the privilege granted by the <ACE> policy given the <SubjectId> and <ResourceRef> matching does not produce the empty set match.

Permissions are defined in terms of CREATE ("C"), RETRIEVE ("R"), UPDATE ("U"), DELETE ("D"), NOTIFY ("N") and NIL ("-"). NIL is substituted for a permissions character that signifies the respective permission is not granted.

The empty set match result defaults to a condition where no access rights are granted.

If the <Validity> token exists, the <Permission> granted is constrained to the time <Period>. <Validity> may further be segmented into a <Recurrence> pattern where access may alternatively be granted and rescinded according to the pattern.

13.5.3 ACL Resource

An "acl2" is a list of type "ace2".

In order to provide an interface which allows management of array elements of the "aclist2" Property associated with a "/oic/sec/acl2" Resource, the RETRIEVE, UPDATE and DELETE operations on the" /oic/sec/acl2" Resource SHALL behave as follows:

1) A RETRIEVE shall return the full Resource representation.

2) An UPDATE shall replace or add to the Properties included in the representation sent with the UPDATE request, as follows:

   a) If an UPDATE representation includes the "aclist2" array Property, then:

      i) Supplied ACEs with an "aceid" that matches an existing "aceid" shall replace completely the corresponding ACE in the existing "aclist2" array.

      ii) Supplied ACEs without an "aceid" shall be appended to the existing "aclist2" array, and a unique (to the "/oic/sec/acl2" Resource) "aceid" shall be created and assigned to the new ACE by the Server. The "aceid" of a deleted ACE should not be reused, to improve the determinism of the interface and reduce opportunity for race conditions.

      iii) Supplied ACEs with an "aceid" that does not match an existing "aceid" shall be appended to the existing "aclist2" array, using the supplied "aceid".

      iv) The rows in Table 34 corresponding to the "aclist2" array Property dictate the Device States in which an UPDATE of the "aclist2" array Property is always rejected. If OCF Device is in a Device State where the Access Mode in this row contains "R", then the OCF Device shall reject all UPDATEs of the "aclist2" array Property.

3) A DELETE without query parameters shall set the "aclist2" array to the empty array, but shall not remove the "oic/sec/ace2" Resource.

4) A DELETE with one or more "aceid" query parameters shall remove the ACE(s) with the corresponding "aceid"(s) from the "aclist2" array.

5) The rows in Table 34 corresponding to the "aclist2" array Property dictate the Device States in which a DELETE is always rejected. If OCF Device is in a Device State where the Access Mode in this row contains "R", then the OCF Device shall reject all DELETEs.

NOTE The "/oic/sec/acl2" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined in ISO/IEC 30118-1.

Evaluation of local ACL Resource completes when all ACL Resource have been queried and no entry can be found for the requested Resource for the requestor – e.g. "/oic/sec/acl2" does not match the subject and the requested Resource.

Table 32 defines the values of "oic.sec.crudntype".

Table 32 – Value definition of the "oic.sec.crudntype" Property

Value	Access Policy	Description	RemarksNotes
bx0000,0000 (0)	No permissions	No permissions	N/A
bx0000,0001 (1)	C	CREATE	N/A
bx0000,0010 (2)	R	RETREIVE, OBSERVE, DISCOVER	The "R" permission bit covers both the Read permission and the Observe permission.
bx0000,0100 (4)	U	WRITE, UPDATE	N/A
bx0000,1000 (8)	D	DELETE	N/A
bx0001,0000 (16)	N	NOTIFY	The "N" permission bit is ignored in OCF 1.0, since "R" covers the Observe permission. It is documented for future versions

"oic/sec/acl2" Resource is defined in Table 20.

Table 33 – Definition of the "oic/sec/acl2" Resource

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/acl2	ACL2	oic.r.acl2	oic.if.baseline, oic.if.rw	Resource for managing access	Security

Table 34 defines the Properties of "oic.sec.acl2".

Table 34 – Properties of the "/oic/sec/acl2" Resource

Property Name	Value Type	Mandatory	Device State	Access Mode	Description
aclist2	array of oic.sec.ace2	Yes	N/A		The aclist2 Property is an array of ACE records of type "oic.sec.ace2". The Server uses this list to apply access control to its local Resources.
N/A	N/A	N/A	RESET	R	Server shall set to manufacturer defaults.
			RFOTM	RW	Set by DOTS after successful OTM
			RFPRO	RW	The AMS (referenced via rowneruuid property) shall update the aclist entries after mutually authenticated secure session is established. Access to NCRs is prohibited.
			RFNOP	R	Access to NCRs is permitted after a matching ACE2 is found.
			SRESET	RW	The DOTS (referenced via devowneruuid Property of "/oic/sec/doxm Resource") should evaluate the integrity of and may update aclist entries when a secure session is established and the Server and DOTS are authenticated.
rowneruuid	uuid	Yes	N/A		The Resource owner Property (rowneruuid) is used by the Server to reference a service provider trusted by the Server. Server shall verify the service provider is authorized to perform the requested action
			RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
			RFOTM	RW	The DOTS should configure the rowneruuid Property of "/oic/sec/acl2" Resource when a successful owner transfer session is established.
			RFPRO	R	n/a
			RFNOP	R	n/a
			SRESET	RW	The DOTS (referenced via devowneruuid Property or rowneruuid Property of "/oic/sec/doxm" Resource) should verify and if needed, update the Resource owner Property when a mutually authenticated secure session is established. If the rowneruuid Property does not refer to a valid DOTS the Server shall transition to RESET.

Table 35 defines the Properties of "oic.sec.ace2".

Table 35 – "oic.sec.ace2" data type definition.

Property Name	Value Type	Mandatory	Description
subject	oic.sec.roletype, oic.sec.didtype, oic.sec.conntype	Yes	The Client is the subject of the ACE when the roles, Device UUID, or connection type matches.
resources	array of oic.sec.ace2.resource-ref	Yes	The application's Resources to which a security policy applies
permission	oic.sec.crudntype.bitmask	Yes	Bitmask encoding of CRUDN permission
validity	array of oic.sec.time-pattern	No	An array of a tuple of period and recurrence. Each item in this array contains a string representing a period using the IETF RFC 5545 Period, and a string array representing a recurrence rule using the IETF RFC 5545 Recurrence.
aceid	integer	Yes	An aceid is unique with respect to the array entries in the aclist2 Property.

Table 36 defines the Properties of "oic.sec.ace2.resource-ref ".

Table 36 – "oic.sec.ace2.resource-ref" data type definition.

Property Name	Value Type	Mandatory	Description
href	uri	No	A URI referring to a Resource to which the containing ACE applies
wc	string	No	Refer to Table 15.

Table 37 defines the values of "oic.sec.ace2.resource-ref ".

Table 37 – Value definition "oic.sec.conntype" Property

Property Name	Value Type	Value Rule	Description
conntype	string	enum ["auth-crypt", "anon-clear"]	This Property allows an ACE to be matched based on the connection or message protection type
		auth-crypt	ACE applies if the Client is authenticated and the data channel or message is encrypted and integrity protected
		anon-clear	ACE applies if the Client is not authenticated and the data channel or message is not encrypted but may be integrity protected

Local ACL Resources supply policy to a Resource access enforcement point within an OCF stack instance. The OCF framework gates Client access to Server Resources. It evaluates the subject's request using policies contained in ACL Resources.

Resources named in the ACL policy can be fully qualified or partially qualified. Fully qualified Resource references include the device identifier in the href Property that identifies the remote Resource Server that hosts the Resource. Partially qualified references mean that the local Resource Server hosts the Resource. If a fully qualified Resource reference is given, the Intermediary enforcing access shall have a secure channel to the Resource Server and the Resource Server shall verify the Intermediary is authorized to act on its behalf as a Resource access enforcement point.

Resource Servers should include references to Device and ACL Resources where access enforcement is to be applied. However, access enforcement logic shall not depend on these references for access control processing as access to Server Resources will have already been granted.

Local ACL Resources identify a Resource Owner service that is authorized to instantiate and modify this Resource. This prevents non-terminating dependency on some other ACL Resource. Nevertheless, it should be desirable to grant access rights to ACL Resources using an ACL Resource.

An ACE2 entry is considered "currently valid" if the validity period of the ACE2 entry includes the time of the request. The validity period in the ACE2 may be a recurring time period (e.g., daily from 1:00-2:00). Matching the Resource(s) specified in a request to the "resource" Property of the ACE2 is defined in clause 12.2. For example, one way they can match is if the Resource URI in the request exactly matches one of the Resource references in the ACE2 entries.

A request will match an ACE2 if any of the following are true:

1) The ACE2 "subject" Property is of type "oic.sec.didtype" has a UUID value that matches the "deviceuuid" Property associated with the secure session;

   AND the Resource of the request matches one of the "resources" Property of the ACE2 "oic.sec.ace2.resource-ref";

   AND the ACE2 is currently valid.

2) The ACE2 "subject" Property is of type "oic.sec.conntype" and has the wildcard value that matches the currently established connection type;

   AND the Resource of the request matches one of the "resources" Property of the ACE2 "oic.sec.ace2.resource-ref";

   AND the ACE2 is currently valid.

3) When Client authentication uses a certificate credential;

   AND one of the "roleid" values contained in the role certificate matches the "roleid" Property of the ACE2 "oic.sec.roletype";

   AND the role certificate public key matches the public key of the certificate used to establish the current secure session;

   AND the Resource of the request matches one of the array elements of the "resources" Property of the ACE2 "oic.sec.ace2.resource-ref";

   AND the ACE2 is currently valid.

4) When Client authentication uses a certificate credential;

   AND the CoAP payload query string of the request specifies a role, which is member of the set of roles contained in the role certificate;

   AND the roleid values contained in the role certificate matches the "roleid" Property of the ACE2 "oic.sec.roletype";

   AND the role certificate public key matches the public key of the certificate used to establish the current secure session;

   AND the Resource of the request matches one of the "resources" Property of the ACE2 "oic.sec.ace2.resource-ref";

   AND the ACE2 is currently valid.

5) When Client authentication uses a symmetric key credential;

   AND one of the "roleid" values associated with the symmetric key credential used in the secure session, matches the "roleid" Property of the ACE2 "oic.sec.roletype";

AND the Resource of the request matches one of the array elements of the "resources" Property of the ACE2 "oic.sec.ace2.resource-ref";

AND the ACE2 is currently valid.

6) When Client authentication uses a symmetric key credential;

AND the CoAP payload query string of the request specifies a role, which is contained in the "oic.r.cred.creds.roleid" Property of the current secure session;

AND CoAP payload query string of the request specifies a role that matches the "roleid" Property of the ACE2 "oic.sec.roletype";

AND the Resource of the request matches one of the array elements of the "resources" Property of the ACE2 "oic.sec.ace2.resource-ref";

AND the ACE2 is currently valid.

A request is granted if ANY of the ‘matching’ ACE2 entries contain the permission to allow the request. Otherwise, the request is denied.

There is no way for an ACE2 entry to explicitly deny permission to a Resource. Therefore, if one Device with a given role should have slightly different permissions than another Device with the same role, they must be provisioned with different roles.

The Server is required to verify that any hosted Resource has authorized access by the Client requesting access. The "/oic/sec/acl2" Resource is co-located on the Resource host so that the Resource request processing should be applied securely and efficiently. See Annex A for an example.

13.6 Access Manager ACL Resource [Deprecated]

This clause is intentionally left blank.

13.7 Signed ACL Resource [Deprecated]

This clause is intentionally left blank.

13.8 Provisioning Status Resource

The "/oic/sec/pstat" Resource maintains the Device provisioning status. Device provisioning should be Client-directed or Server-directed. Client-directed provisioning relies on a Client device to determine what, how and when Server Resources should be instantiated and updated. Server-directed provisioning relies on the Server to seek provisioning when conditions dictate. Furthermore, the "/oic/sec/cred" Resource should be provisioned at ownership transfer with credentials necessary to open a secure connection with appropriate support service.

"/oic/sec/pstat" Resource is defined in Table 38.

Table 38 – Definition of the "/oic/sec/pstat" Resource

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/pstat	Provisioning Status	oic.r.pstat	oic.if.baseline, oic.if.rw	Resource for managing Device provisioning status	Configuration

Table 39 defines the Properties of "/oic/sec/pstat".

Table 39 – Properties of the "/oic/sec/pstat" Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Device Onboarding State	dos	oic.sec.dostype	N/A	Yes	RW		Device Onboarding State
Is Device Operational	isop	Boolean	T|F	Yes	R	RESET	Server shall set to FALSE
					R	RFOTM	Server shall set to FALSE
					R	RFPRO	Server shall set to FALSE
					R	RFNOP	Server shall set to TRUE
					R	SRESET	Server shall set to FALSE
Current Mode	cm	oic.sec.dpmtype	bitmask	Yes	R		Current Mode
Target Mode	tm	oic.sec.dpmtype	bitmask	Yes	RW		Target Mode
Operational Mode	om	oic.sec.pomtype	bitmask	Yes	R	RESET	Server shall set to manufacturer default.
					RW	RFOTM	Set by DOTS after successful OTM
					RW	RFPRO	Set by CMS, AMS, DOTS after successful authentication
					RW	RFNOP	Set by CMS, AMS, DOTS after successful authentication
					RW	SRESET	Set by DOTS.
Supported Mode	sm	oic.sec.pomtype	bitmask	Yes	R	All states	Supported provisioning services operation modes
Device UUID	deviceuuid	String	uuid	Yes	RW	All states	[DEPRECATED] A uuid that identifies the Device to which the status applies
Resource Owner ID	rowneruuid	String	uuid	Yes	R	RESET	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
					RW	RFOTM	The DOTS should configure the rowneruuid Property when a successful owner transfer session is established.
					R	RFPRO	n/a
					R	RFNOP	n/a
					RW	SRESET	The DOTS (referenced via devowneruuid Property of "/oic/sec/doxm" Resource) should verify and if needed, update the Resource owner Property when a mutually authenticated secure session is established. If the rowneruuid does not refer to a valid DOTS the Server shall transition to RESET.

Table 40 defines the Properties of "oic.sec.dostype".

Table 40 – Properties of the ".oic.sec.dostype" Property

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Device Onboarding State	s	UINT16	enum (0=RESET, 1=RFOTM, 2=RFPRO, 3=RFNOP, 4=SRESET	Y	R	RESET	The Device is in a hard reset state.
					RW	RFOTM	Set by DOTS after successful OTM to RFPRO.
					RW	RFPRO	Set by CMS, AMS, DOTS after successful authentication
					RW	RFNOP	Set by CMS, AMS, DOTS after successful authentication
					RW	SRESET	Set by CMS, AMS, DOTS after successful authentication
Pending state	p	Boolean	T | F	Y	R	All States	FALSE (0) – "s" state changes are complete. Since Device is not able to respond when the value is TRUE, other values of this property are DEPRECATED.

In all Device states:

- The Device permits an authenticated and authorised Client to change the Device state of a Device by updating the "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource to the desired value. The allowed Device state transitions are defined in Figure 22.
- Prior to updating the "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource, the Client configures the Device to meet entry conditions for the new Device state. The SVR definitions define the entity (Client or Server) expected to perform the specific SVR configuration change to meet the entry conditions. Once the Client has configured the aspects for which the Client is responsible, it can update the "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource. The Server then makes any changes for which the Server is responsible, including updating required SVR values, and set the "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource to the new value.

When Device state is RESET:

- All SVR content is removed and reset to manufacturer default values.
- The default manufacturer Device state is RESET.
- NCRs are reset to manufacturer default values.
- NCRs shall not be accessible.
- After successfully processing RESET the SRM transitions to RFOTM by setting the "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource to 1 (RFOTM).

When Device state is RFOTM:

- NCRs shall not be accessible.
- Before OTM is successful, the the "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource is read-only by unauthenticated requestors
- After the OTM is successful, the "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource is read-write by authorized requestors.
- The negotiated Device OC is used to create an authenticated session over which the DOTS directs the Device state to transition to RFPRO.

- If an authenticated session cannot be established the ownership transfer session should be disconnected and SRM sets back the Device state to RESET.
- Ownership transfer session, especially Random PIN OTM, should not exceed 60 seconds. If the SRM asserts the OTM failed, the ownership transfer session should be disconnected, and the Device should transition to RESET ("/pstat.dos.s"=0 (RESET)).
- The DOTS UPDATES the "devowneruuid" Property in the "/oic/sec/doxm" Resource to a non-nil UUID value. The DOTS (or other authorized client) can update it multiple times while in RFOTM. It is not updatable while in other device states except when the Device state returns to RFOTM through RESET.
- The DOTS can have additional provisioning tasks to perform while in RFOTM. When done, the DOTS UPDATES the "owned" Property in the "/oic/sec/doxm" Resource to "true".
- After successful OTM, the DOTS triggers the transition to RFPRO and the "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource is set to 2 (RFPRO).

When Device state is RFPRO:

- The "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource is read-only by unauthorized requestors and read-write by authorized requestors.
- NCRs shall not be accessible, except for Easy Setup Resources, if supported.
- An authorized Client may provision SVRs as needed for normal functioning in RFNOP.
- An authorized Client may perform consistency checks on SVRs to determine which shall be re-provisioned.
- Failure to successfully provision SVRs may trigger a state change to RESET. For example, if the Device has already transitioned from SRESET but consistency checks continue to fail.
- The authorized Client sets the "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource to 3 (RFNOP).

When Device state is RFNOP:

- The "s" Property of the "dos" Property of the "/oic/sec/pstat" Resource is read-only by unauthorized requestors and read-write by authorized requestors.
- NCRs, SVRs and core Resources are accessible following normal access processing.
- When additional provisioning is necessary, the Device may be transitioned to RFPRO by an authorized Client. Only the Device owner should transition to SRESET or RESET.

When Device state is SRESET:

- NCRs shall not be accessible. The integrity of NCRs may be suspect but the SRM doesn’t attempt to access or reference them.
- SVR integrity is not guaranteed, but access to some SVR Properties is necessary. These include "devowneruuid" Property of the "/oic/sec/doxm" Resource, "creds":[{…,{"subjectuuid":<devowneruuid>},…}] Property of the "/oic/sec/cred" Resource and "pstat.dos.s" "/oic/sec/pstat" Resource.
- The certificates that identify and authorize the Device owner are sufficient to re-create minimalist "/oic/sec/cred" and "/oic/sec/doxm" Resources enabling Device owner control of SRESET. If the SRM can’t establish these Resources, then it will transition to RESET.
- An authorized Client performs SVR consistency checks. The authorized Client can provision SVRs as needed to ensure they are available for continued provisioning in RFPRO or for normal functioning in RFNOP.
- The authorized Device owner can avoid entering RESET and RFOTM by UPDATING "pstat.dos.s" with RFPRO or RFNOP values.

- ACLs on SVR are presumed to be invalid. Access authorization is granted according to Device owner privileges only.
- The SRM asserts a Client-directed operational mode (e.g. "/pstat.om"=4).

The provisioning mode type is a 16-bit mask enumerating the various Device provisioning modes. "{ProvisioningMode}" should be used in this document to refer to an instance of a provisioning mode without selecting any particular value.

"oic.sec.dpmtype" is defined in Table 41.

Table 41 – Definition of the "oic.sec.dpmtype" Property

Type Name	Type URN	Description
Device Provisioning Mode	oic.sec.dpmtype	Device provisioning mode is a 16-bit bitmask describing various provisioning modes

Table 42 and Table 43 define the values of "oic.sec.dpmtype".

Table 42 – Value Definition of the "oic.sec.dpmtype" Property (Low-Byte)

Value	Device Mode	Description
bx0000,0001 (1)	Deprecated	
bx0000,0010 (2)	Deprecated	
bx0000,0100 (4)	Deprecated	
bx0000,1000 (8)	Deprecated	
bx0001,0000 (16)	Deprecated	
bx0010,0000 (32)	Deprecated	
bx0100,0000 (64)	Initiate Software Version Validation	Software version validation requested/pending (1) Software version validation complete (0) Requires software download to verify integrity of software package
bx1000,0000 (128)	Initiate Secure Software Update	Secure software update requested/pending (1) Secure software update complete (0)

Table 43 – Value Definition of the "oic.sec.dpmtype" Property (High-Byte)

Value	Device Mode	Description
bx0000,0001 (1)	Initiate Software Availability Check	Checks if new software is available on remote endpoint. Does not require to download software. Methods used are out of bound.
Bits 2-8	<Reserved>	Reserved for later use

The provisioning operation mode type is an 8-bit mask enumerating the various provisioning operation modes.

"oic.sec.pomtype" is defined in Table 44.

Table 44 – Definition of the "oic.sec.pomtype" Property

Type Name	Type URN	Description
Device Provisioning OperationMode	oic.sec.pomtype	Device provisioning operation mode is a 8-bit bitmask describing various provisioning operation modes

Table 45 defines the values of "oic.sec.pomtype".

Table 45 – Value Definition of the "oic.sec.pomtype" Property

Value	Operation Mode	Description
bx0000,0001 (1)	Server-directed utilizing multiple provisioning services	Deprecated
bx0000,0010 (2)	Server-directed utilizing a single provisioning service	Deprecated
bx0000,0100 (4)	Client-directed provisioning	Device supports provisioning service control of this Device’s provisioning operations. This bit is always TRUE.
bx0000,1000(8) – bx1000,0000(128)	<Reserved>	Reserved for later use
bx1111,11xx	<Reserved>	Reserved for later use

13.9 Certificate Signing Request Resource

The "/oic/sec/csr" Resource is used by a Device to provide its desired identity, public key to be certified, and a proof of possession of the corresponding private key in the form of a IETF RFC 2986 PKCS#10 Certification Request. If the Device supports certificates (i.e. the "sct" Property of "/oic/sec/doxm" Resource has a 1 in the 0x8 bit position), the Device shall have a "/oic/sec/csr" Resource.

"/oic/sec/csr" Resource is defined in Table 46.

Table 46 – Definition of the "/oic/sec/csr" Resource

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/csr	Certificate Signing Request	oic.r.csr	oic.if.baseline, oic.if.rw	The CSR Resource contains a Certificate Signing Request for the Device’s public key.	Configuration

Table 47 defines the Properties of "/oic/sec/csr ".

Table 47 – Properties of the "oic.r.csr" Resource

Property Title	Property Name	Value Type	Access Mode	Mandatory	Description
Certificate Signing Request	csr	String	R	Yes	Contains the signed CSR encoded according to the encoding Property
Encoding	encoding	String	R	Yes	A string specifying the encoding format of the data contained in the csr Property "oic.sec.encoding.pem" – Encoding for PEM-encoded certificate signing request

The Device chooses which public key to use, and may optionally generate a new key pair for this purpose.

In the CSR, the Common Name component of the Subject Name shall contain a string of the format "uuid:X" where X is the Device’s requested UUID in the format defined by IETF RFC 4122. The Common Name, and other components of the Subject Name, may contain other data. If the Device chooses to include additional information in the Common Name component, it shall delimit it from the UUID field by white space, a comma, or a semicolon.

If the Device does not have a pre-provisioned key pair to use, but is capable and willing to generate a new key pair, the Device may begin generation of a key pair as a result of a RETRIEVE of this

Resource. If the Device cannot immediately respond to the RETRIEVE request due to time required to generate a key pair, the Device shall return an "operation pending" error. This indicates to the Client that the Device is not yet ready to respond, but will be able at a later time. The Client should retry the request after a short delay.

13.10 Roles Resource

The "roles" Resource maintains roles that have been asserted with role certificates, as described in clause 10.4.2. Asserted roles have an associated public key, i.e., the public key in the role certificate. Servers shall only grant access to the roles information associated with the public key of the Client. The roles Resource should be viewed as an extension of the (D)TLS session state. See 10.4.2 for how role certificates are validated.

The roles Resource shall be created by the Server upon establishment of a secure (D)TLS session with a Client, if is not already created. The roles Resource shall only expose a secured OCF Endpoint in the "/oic/res" response. A Server shall retain the roles Resource at least as long as the (D)TLS session exists. A Server shall retain each certificate in the roles Resource at least until the certificate expires or the (D)TLS session ends, whichever is sooner. The requirements of clause 10.3 and 10.4.2 to validate a certificate's time validity at the point of use always apply. A Server should regularly inspect the contents of the roles Resource and purge contents based on a policy it determines based on its resource constraints. For example, expired certificates, and certificates from Clients that have not been heard from for some arbitrary period of time could be candidates for purging.

The OCF namespace ("oic.role.*") is restricted to OCF-defined roles. "oic.role.owner" is an OCF-defined Role that is intended to provide Resource Owner privileges to multiple Clients in a scalable way. Servers shall grant access to perform all supported operations in the current Device state (see clause 8) on all supported SVRs regardless of ACL configuration the Clients asserting "oic.role.owner" Role. Servers shall reject assertion of any Role, which starts with "oic.role.", but is not one of the following Roles:

– "oic.role.owner"

The "roles" Resource is implicitly created by the Server upon establishment of a (D)TLS session. In more detail, the RETRIEVE, UPDATE and DELETE operations on the roles Resource shall behave as follows. Unlisted operations are implementation specific and not reliable.

1) A RETRIEVE request shall return all previously asserted roles associated with the currently connected and authenticated Client's identity. RETRIEVE requests with a "credid" query parameter is not supported; all previously asserted roles associated with the currently connected and authenticated Client's identity are returned.
2) An UPDATE request that includes the "roles" Property shall replace or add to the Properties included in the array as follows:
   a) If either the "publicdata" or the "optionaldata" are different than the existing entries in the "roles" array, the entry shall be added to the "roles" array with a new, unique "credid" value.
   b) If both the "publicdata" and the "optionaldata" match an existing entry in the "roles" array, the entry shall be considered to be the same. The Server shall reply with a 2.04 Changed response and a duplicate entry shall not be added to the array.
   c) The "credid" Property is optional in an UPDATE request and if included, it may be ignored by the Server. The Server shall assign a unique "credid" value for every entry of the "roles" array.
3) A DELETE request without a "credid" query parameter shall remove all entries from the "/oic/sec/roles" Resource array corresponding to the currently connected and authenticated Client's identity.

4) A DELETE request with a "credid" query parameter shall remove only the entries of the "/oic/sec/roles" Resource array corresponding to the currently connected and authenticated Client's identity and where the corresponding "credid" matches the entry.

NOTE The "/oic/sec/roles" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined in ISO/IEC 30118-1.

See clause 8 for restrictions on the states in which this Resource may be modified.

"/oic/sec/roles" Resource is defined in Table 48.

Table 48 – Definition of the "/oic/sec/roles" Resource

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/roles	Roles	oic.r.roles	oic.if.baseline, oic.if.rw	Resource containing roles that have previously been asserted to this Server	Security

Table 49 defines the Properties of "/oic/sec/roles".

Table 49 – Properties of the "/oic/sec/roles" Resource

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Roles	roles	oic.sec.cred	array	RW	Yes	List of roles previously asserted to this Server

Because "/oic/sec/roles" shares the "oic.sec.cred" schema with "/oic/sec/cred", "subjectuuid" is a required Property. However, "subjectuuid" is not used in a role certificate. Therefore, a Device may ignore the "subjectuuid" Property if the Property is contained in an UPDATE request to the "/oic/sec/roles" Resource.

13.11 Auditable Events List Resource

13.11.1 Auditable Events List Resource general

The "/oic/sec/ael" Resource maintains a list of logged Auditable Events. Every OCF Device logs AEEs filtered according to the values of the "categoryfilter" and "priorityfilter" Properties of "/oic/sec/ael" Resource. All Devices shall have a "/oic/sec/ael" Resource to maintain AEEs. The new AEE shall be added to the "events" Property of "/oic/sec/ael" Resource as the last entry in the array. A Device shall store all AEEs of the "/oic/sec/ael" Resource in non-volatile memory. A Device shall be able to store at least 1 AEE.

The "categoryfilter" Property determines what categories of AEEs are to be logged. The "categoryfilter" Property is an integer value which is a composition of bitmasks. A Device shall log all AEEs filtered by this value. If the "categoryfilter" is either set to 0xff or is not set, then the Device shall log AEEs of all categories. Refer to Table 51 for more details.

The "priorityfilter" Property determines the lowest priority of AEE to be logged. A smaller value means higher priority. The AEEs whose "priority" Property values are equal to or smaller than this value shall be logged. If the "priorityfilter" Property is either set to the highest priority or is not set, then the Device shall log all AEEs. No matter what value is set to "priorityfilter", an AEE of CRIT (== 0) "priority" shall always be logged. Refer to Table 51 for more details.

When an AEE is added, the "usedspace" Property shall be updated to reflect the total storage used by all logged events. When the reserved storage for AEEs is full, the oldest AEE shall be purged.

A Device logs a new AEE as follows:

5) If a new AEE is not filtered by "categoryfilter" and "priorityfilter", then it is dropped.

```
/* c-like pseudo code */
If ((categoryfilter & new_aee->category) && (priorityfilter >= new_aee->priority))
```

```
{
    addAEE(new_aee);
}
else
{
    free(new_aee);
}
```

6) If the value of "usedspace" Property is equal to, or the sum of the "usedspace" Property value and the size of the new AEE is bigger than the value of the "maxspace" Property of "/oic/sec/ael" Resource, then:

   a) Remove the oldest AEE continuously while the sum of the "usedspace" Property value and the size of the new AEE is bigger than the "maxspace" Property value.

```
/* c-like pseudo code */
Int addAEE(AEEtype *new_aee)
{

    While ((usespace + new_aee->size) > maxspace)
    {
        /* purgeAEE() returns the size of purged AEE */
        sizeOfPurgedAEE = purgeAEE();
        usedspace -= sizeOfPurgedAEE;
    }

    ...
    ...
}
```

7) Add the new AEE to the "events" array Property of the "/oic/sec/ael" Resource as the last entry in the array.

8) Increase the value of the "usedspace" Property by the size of the new AEE.

In order to provide a mechanism which allows management of the "events" array Property, the RETRIEVE and UPDATE operations on the "/oic/sec/ael" Resource shall behave as follows:

9) A RETRIEVE operation shall return the full Resource representation.

10) An UPDATE operation may set the "categoryfilter" and/or "priorityfilter" Properties.

The "/oic/sec/ael" Resource is defined in Table 50.

Table 50 – Definition of the "/oic/sec/ael" Resource

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/ael	Auditable Event List	oic.r.ael	oic.if.baseline, oic.if.rw	Resource for storing AEEs	Security

Table 51 defines the Properties of the "/oic/sec/ael" Resource.

Table 51 – Properties of the "/oic/sec/ael" Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
AEE list	"events"	"array"		Yes	RESET	R	The Device clears

			Array of "oic.sec.aee" entries		RFOTM	R	This list stores AEEs whose "category" Property value is filtered by "categoryfilter" Property and "priority" Property value is equal or less than the value of "priorityfilter" Property.
					RFPRO		
					RFNOP		
					SRESET		
current used storage size	"usedspace"	"integer"	>= 0 (default: 0)	Yes	RESET	R	The Device sets to 0
					RFOTM	R	Current used space for logged AEEs. The Device updates this Property whenever new AEEs are logged.
					RFPRO		
					RFNOP		
					SRESET		
maximum allowed storage size for AEEs	"maxspace"	"integer"	> 0	Yes		R	This means the maximum allowable storage size for AEEs that can be stored in "events" list. The Manufacturer chooses this value.
unit for storage size	"unit"	"string"	enum ["Kbyte", "Byte"] (default: “Byte”)	No		R	The unit for "usedspace" and "maxspace" Properties. The Manufacturer chooses this value.
Categories of AEE to be logged	"categoryfilter"	"integer"	bitmask (default: 0xff)	Yes	RESET	R	The Device sets to the manufacturer default value
					RFOTM	RW	This value decides what categories of AEEs are to be logged. Meaning of each bit: • 0x01 (Access Control) • 0x02 (Onboarding) • 0x04 (Device) • 0x08 (Authentication) • 0x10 (SVR Modification) • 0x20 (Cloud) • 0x40 (Communication) • 0x80 (Reserved) e.g.) if "categoryfilter" == 0xff: log all events of all categories e.g.) if "categoryfilter" == 0x03: log all events of 'AC (== 0x01)' and 'OB (==0x02)' categories
					RFPRO		
					RFNOP	R	
					SRESET	RW	
Minimum priority of AEEs to be logged	"priorityfilter"	"integer"	enum [0, 1, 2, 3, 4] (default: 4)	Yes	RESET	R	Device sets to manufacturer default value
					RFOTM	RW	The AEEs whose "priority" values are equal to or smaller than this value are logged. A smaller value means a higher priority. Meaning of each value: • 0 (CRIT) • 1 (ERR) • 2 (WARN) • 3 (INFO) • 4 (DEBUG) e.g.) if "priorityfilter" is set to DEBUG (==4) all AEEs will be logged
					RFPRO		
					RFNOP	R	
					SRESET	RW	

							e.g.) if "priorityfilter" is set to 1, CRIT (==0) and ERR (==1) SEEs will be logged

Table 52 defines the Properties of the "oic.sec.aee" type.

Table 52 – "oic.sec.aee" data type definition

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Device State	Description
Auditable Event Identifier	"aeid"	"string"	N/A	R	Yes	-	Identity of the logged event
Category of AEE	"category"	"integer"	enum [1, 2, 4, 8, 16, 32, 64, 128]	R	Yes	-	The category of this AEE: • 0x01 (Access Control) • 0x02 (Onboarding) • 0x04 (Device) • 0x08 (Authentication) • 0x10 (SVR Modification) • 0x20 (Cloud) • 0x40 (Communication) • 0x80 (Reserved)
Priority of AEE	"priority"	"integer"	enum [0, 1, 2, 3, 4]	R	Yes	-	The priority of this AEE: • 0 (CRIT) • 1 (ERR) • 2 (WARN) • 3 (INFO) • 4 (DEBUG)
Time stamp	"timestamp"	"string"	date-time (RFC3339 clause 5.6)	R	Yes	-	The time when the AEE occured
Event message	"message"	"string"	N/A	R	Yes	-	The description of the logged AEE.
Auxiliary info	"auxiliaryinfo"	"array"	Array of strings	R	Yes	-	Supplementary information for the "message" Property e.g.) URI of specific Resource in ACE2

OCF-defined AEEs are listed in Table 54, and each such AEE has its own values for the "category" and "priority" Properties.

The "timestamp" Property follows a full-date and partial-time format of RFC3339. Every new AEE shall have a later timestamp than the latest previously logged AEE.

The "auxiliaryinfo" Property provides supplementary info which is not covered by the description in "message" Property. For example, the URI of specific Resource in ACE2 could be "auxiliaryinfo" for "Access Denied" AEE. Please see Table 54 "List of Auditable Events".

13.12 Security Virtual Resources (SVRs) and Access Policy

The SVRs expose the security-related Properties of the Device.

Granting access requests (RETRIEVE, UPDATE, DELETE, etc.) for these SVRs to unauthenticated (anonymous) Clients could create privacy or security concerns.

For example, when the Device onboarding State is RFOTM, it is necessary to grant requests for the "/oic/sec/doxm" Resource to anonymous requesters, so that the Device can be discovered and onboarded by an OBT. Subsequently, it might be preferable to deny requests for the "/oic/sec/doxm" Resource to anonymous requesters, to preserve privacy.

13.13 SVRs, discoverability and OCF Endpoints

All implemented SVRs shall be "discoverable" (reference ISO/IEC 30118-1, Policy Parameter clause 7.8.2.1.2).

All implemented discoverable SVRs shall expose a Secure OCF Endpoint (e.g. CoAPS) (reference ISO/IEC 30118-1, clause 10).

The "/oic/sec/doxm" Resource shall expose an Unsecure OCF Endpoint (e.g. CoAP) in RFOTM (reference ISO/IEC 30118-1, clause 10).

13.14 Additional privacy consideration for Core Resources

Unique immutable identifiers are a privacy consideration due to their potential for being used as a tracking mechanism. These include the following Resources and Properties:

– "/oic/d" Resource containing the "piid" Property.

– "/oic/p" Resource containing the "pi" Property.

These identifiers are unique values that are visible at various times throughout the Device lifecycle by anonymous requestors. This implies any Client Device, including those with malicious intent, are able to reliably obtain identifiers useful for building a log of activity correlated with a specific Platform and Device.

The "di" Property in the "/oic/d" Resource shall mirror that of the "deviceuuid" Property of the "/oic/sec/doxm" Resource. The DOTS should provision an ACL policy that restricts access to the "/oic/d" Resource such that only authenticated Clients are able to obtain the "di" Property of "/oic/d" Resource. See clause 13.1 for deviceuuid Property lifecycle requirements.

Servers should expose a temporary, non-repeated, "piid" Property of "/oic/d" Resource Value upon entering RESET. Servers shall expose a persistent value via the "piid" Property of "/oic/d" Property when the DOTS sets "devowneruuid" Property to a non-nil-UUID value. The DOTS should provision an ACL policy on the "/oic/d" Resource such that only authenticated Clients are able to obtain the "piid" Property of "/oic/d" Resource

Servers should expose a temporary, non-repeated, "pi" Property value upon entering RESET. Servers shall expose a persistent value via the "pi" Property of the "/oic/p" Resource when the DOTS sets "devowneruuid" Property to a non-nil-UUID value. The DOTS should provision an ACL policy on the "/oic/p" Resource such that only authenticated Clients are able to obtain the "pi" Property.

Table 53 depicts Core Resource Properties Access Modes given various Device States.

Table 53 – Core Resource Properties Access Modes given various Device States

Resource Type	Property title	Property name	Value type	Access Mode		Behaviour
oic.wk.p	Platform ID	pi	oic.types-schema.uuid	All States	R	Server exposes a temporary random UUID when in RESET.

oic.wk.d	Permanent Immutable ID	piid	oic.types-schema.uuid	All States	R	Server exposes a temporary random UUID when in RESET.
oic.wk.d	Device Identifier	di	oic.types-schema.uuid	All states	R	/d di mirrors the value contained in "/doxm" "deviceuuid" in all device states.

13.15 Easy Setup Resource Device state

This clause only applies to a new Device that uses Easy Setup for ownership transfer as defined in OCF Wi-Fi Easy Setup. Easy Setup has no impact to new Devices that have a different way of connecting to the network i.e. DOTS and AMS don't use a Soft AP to connect to non-Easy Setup Devices.

Figure 29 shows an example of Soft AP and Easy Setup Resource in different Device states.

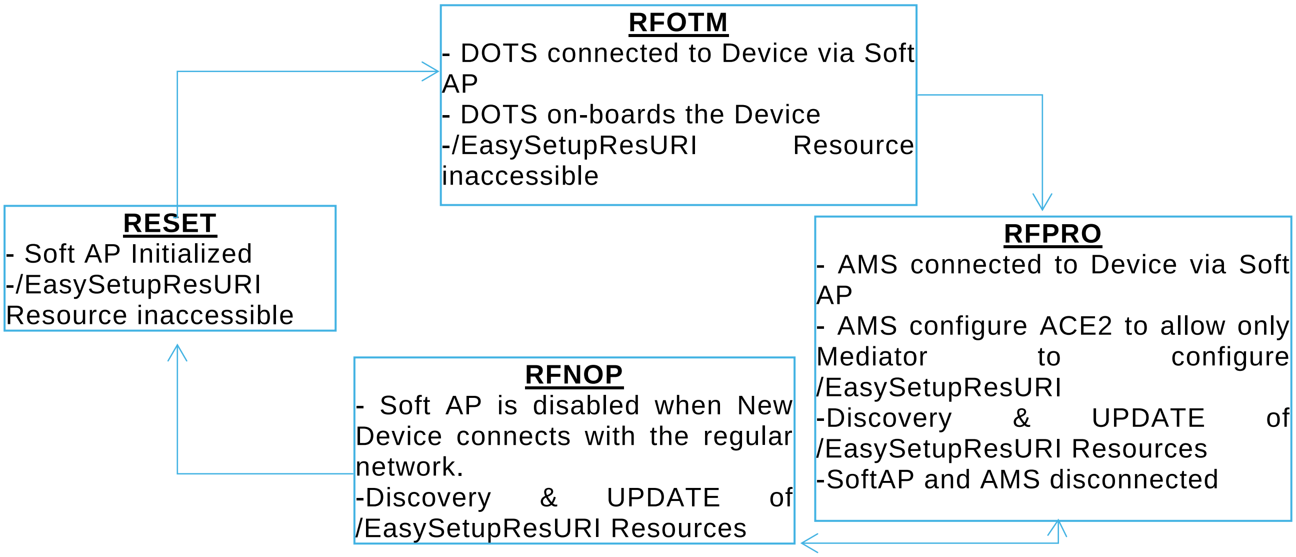

RFOTM
- DOTS connected to Device via Soft AP
- DOTS on-boards the Device
-/EasySetupResURI Resource inaccessible
RESET
- Soft AP Initialized
-/EasySetupResURI Resource inaccessible
RFPRO
- AMS connected to Device via Soft AP
- AMS configure ACE2 to allow only Mediator to configure /EasySetupResURI
-Discovery & UPDATE of /EasySetupResURI Resources
-SoftAP and AMS disconnected
RFNOP
- Soft AP is disabled when New Device connects with the regular network.
-Discovery & UPDATE of /EasySetupResURI Resources

Figure 29 – Example of Soft AP and Easy Setup Resource in different Device states

Device enters RFOTM, Soft AP may be accessible in RFOTM and RFPRO.

While it is reasonable for an End User to expect that power cycling a new Device will turn on the Soft AP for Easy Setup during the initial setup, since that is potentially how it behaved on first boot, it is a security risk to make this the default behaviour of a device that remains unenrolled beyond a reasonable period after first boot.

Therefore, the Soft AP for Easy Setup has several requirements to improve security:

– Time availability of Easy Setup Soft AP should be minimised, and shall not exceed 30 minutes after Device factory reset, RESET or first power boot, or when an End User initiates the Soft AP for Easy Setup.

– If a new Device tried and failed to complete Easy Setup Enrolment immediately following the first boot, or after a factory reset, it may turn the Easy Setup Soft AP back on automatically for another 30 minutes upon being power cycled, provided that the power cycle occurs within 3 hours of first boot or the most recent factory reset. If the End User has initiated the Easy Setup Soft AP directly without a factory reset, it is not necessary to turn it back on if it was on immediately prior to power cycle, because the End User obviously knows how to initiate the process manually.

– After 3 hours from first boot or factory reset without successfully enrolling the device, the Soft AP should not turn back on for Easy Setup until another factory reset occurs, or the End User initiates the Easy Setup Soft AP directly.

– Easy Setup Soft AP may stay enabled during RFNOP, until the Mediator instructs the new Device to connect to the Enroller.

– The Easy Setup Soft AP shall be disabled when the new Device successfully connects to the Enroller.

– Once a new Device has successfully connected to the Enroller, it shall not turn the Easy Setup Soft AP back on for Easy Setup Enrolment again unless the Device is factory reset, or the End User initiates the Easy Setup Soft AP directly.

– Just Works OTM shall not be enabled on Devices which support Easy Setup.

– The Soft AP shall be secured (e.g. shall not expose an open AP).

– The Soft AP shall support a passphrase for connection by the Mediator, and the passphrase shall be between and 8 and 64 ASCII printable characters. The passphrase may be printed on a label, sticker, packaging etc., and may be entered by the End User into the Mediator device.

– The Soft AP should not use a common passphrase across multiple Devices. Instead, the passphrase may be sufficiently unique per device, to prevent guessing of the passphrase by an attacker with knowledge of the Device type, model, manufacturer, or any other information discoverable through Device’s exposed interfaces.

The Enrollee shall support WPA2 security (i.e. shall list WPA2 in the "swat" Property of the "/example/WiFiConfResURI" Resource), for potential selection by the Mediator in connecting the Enrollee to the Enroller. The Mediator should select the best security available on the Enroller, for use in connecting the Enrollee to the Enroller.

The Enrollee may not expose any interfaces (e.g. web server, debug port, NCRs, etc.) over the Soft AP, other than SVRs, and Resources required for Wi-Fi Easy Setup.

The "/example/EasySetupResURI" Resource should not be discoverable in RFOTM or SRESET. After ownership transfer process is completed with the DOTS, and the Device enters in RFPRO, the "/example/EasySetupResURI" may be Discoverable.

The OTM CoAPS session may be used by Mediator for connection over Soft AP for ownership transfer and initial Easy Setup provisioning. SoftAP or regular network connection may be used by AMS for "/oic/sec/acl2" Resource provisioning in RFPRO. The CoAPS session authentication and encryption is already defined in the Security spec.

In RFPRO, AMS is expected to configure ACL2 Resource on the Device with ACE2 for following Resources to be only configurable by the Mediator with permission to UPDATE or RETRIEVE access:

– "/example/EasySetupResURI"

– "/example/WifiConfResURI"

– "/example/DevConfResURI"

An ACE2 granting RETRIEVE or UPDATE access to the Easy Setup Resource

```
{
        "subject": { "uuid": "<insert-UUID-of-Mediator>" },
        "resources": [
            { "href": "/example/EasySetupResURI" },
            { "href": "/example/WiFiConfResURI" },
            { "href": "/example/DevConfResURI" },
        ],
        "permission": 6 // RETRIEVE (2) or UPDATE and RETRIEVE(6)
}
```

ACE2 may be re-configured after Easy Setup process. These ACE2s should be installed prior to the Mediator performing any RETRIEVE/UPDATE operations on these Resources.

In RFPRO or RFNOP, the Mediator should discover /EasySetupResURI Resources and UPDATE these Resources. The Mediator may UPDATE /EasySetupResURI Resources in RFNOP Device state.

A Mediator shall be hosted on an OCF Device.

13.16 List of Auditable Events

Whenever a Device detects an occurrence of any of the Auditable Events in Table 54, then the Device shall log an AEE using the corresponding "category", "priority" and "auxiliaryinfo" Properties defined in Table 54. The "auxiliaryinfo" Property shall contain the entries in the "auxiliaryinfo" column of Table 54 in the order specified in the table with each bullet contained in a separate array entry. The "auxiliaryinfo" Property may contain additional entries for further information following the entries for mandatory information. The "aeid" Property shall include the corresponding Auditable Event Identifier from Table 54.

Table 54 – List of mandatory Auditable Events and corresponding Property values

Auditable Event Identifier ("aeid")	Auditable Event Description	Example "message"	"category"	"priority"	"auxiliaryinfo"
AC-1	A Device received a request from an authenticated Client with valid URI path, valid interface and valid operation for that Resource, but for which access was denied.	“Access Denied”	0x01 (Access Control)	2 (WARN)	• Client IP address & port in format [xxxx:..:xxxx]:xxxx • Client UUID in UUID format (e.g. "00000000-0000-0000-0000-000000000000") • Resource URI (e.g. "/oic/sec/ael") • Requested CRUDN operation (e.g. "CREATE") • Server security state (e.g. "RFNOP") • Asserted roles by Client (e.g. "oic.role.owner"), or "No roles asserted" if there are none
AUTH-1	The Device encountered an error during a DTLS handshaking procedure due to a credential validation failure.	“DTLS handshake failed due to a credential validation failure”	0x08 (Authentication)	1 (ERR)	• Client IP address & port in format [xxxx:..:xxxx]:xxxx •
COMM-1	The Device received a CoAP request which contained unexpected /unsupported CoAP header parameters or unexpected/unsupported CoAP options.	“Unexpected CoAP Command”	0x40 (COMM)	2 (WARN)	• Client IP address & port in format [xxxx:..:xxxx]:xxxx • Hex-encoded CoAP header in format [xx:xx:xx:xx] • Hex-encoded CoAP options except payload (empty if not present)

Whenever a Device detects an occurrence of any of the Auditable Events in Table 55, then the Device should log an AEE using the corresponding "category", "priority" and "auxiliaryinfo" Properties defined in Table 55. The "auxiliaryinfo" Property shall contain the entries in the "auxiliaryinfo" column of Table 55 in the order specified in the table with each bullet contained in a separate array entry. The "auxiliaryinfo" Property may contain additional entries for further information following the entries for mandatory information. The "aeid" Property shall include the corresponding Auditable Event Identifier from Table 55.

Table 55 – List of recommended Auditable Events and corresponding Property values

Auditable Event Identifier	Auditable Event Description	Example "message"	"category"	"priority"	"auxiliaryinfo"
SVR-1	The Device's attempted to use one of its credentials, and detected that the credential is expired	“My credential is expired”	0x10 (SVR Modification)	2 (WARN)	• credid • Credential expiration value
SVR-2	The Device could not validate the role certificate being asserted	"Role assertion failed"	0x10 (SVR Modification)	2 (WARN)	• Client IP address & port in format [xxxx:..:xxxx]:xxxx

13.17 Security Domain Information Resource

The "/oic/sec/sdi" Resource contains the information that identifies the OCF Security Domain to which the Device belongs. OCF Security Domains are uniquely identifiable.

This Resource is optional to implement. When it is exposed by a Device, an OCF Onboarding Tool (OBT) is expected to provision a random UUID and a Security Domain Name for the OCF Security Domain. These two fields are provisioned to a Device during the onboarding process.

"oic.r.sdi" Resource Type is defined in Table 56.

Table 56 –Definition of the "oic.r.sdi" Resource Type

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
"/oic/sec/sdi"	Security Domain Information	"oic.r.sdi"	"oic.if.baseline" "oic.if.rw"	Resource containing Security Domain information	Configuration

Table 57 defines the Properties of "oic.r.sdi".

Table 57 – Properties of the "oic.r.sdi" Resource Type

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Security Domain UUID	"uuid"	string	"uuid"	Yes	R	RESET	A UUID that identifies the Security Domain, set by DOTS during onboarding.
					RW	RFOTM	
					R	RFPRO	
					R	RFNOP	
					R	SRESET	
Security Domain Name	"name"	string	N/A	Yes	R	RESET	Human-friendly name for the Security Domain, set by DOTS during onboarding.
					RW	RFOTM	
					RW	RFPRO	
					R	RFNOP	
					RW	SRESET	
Privacy Flag	"priv"	boolean	N/A	Yes	R	RESET	Flag to indicate whether the Security Domain Information is copied to "/oic/res", and thus whether it is publicly visible or private.
					RW	RFOTM	
					RW	RFPRO	
					R	RFNOP	
					RW	SRESET	

The purpose of the "priv" Property is to control whether information about a Device’s OCF Security Domain is exposed during multicast discoveries.

If the "priv" Property is set to "false", then the "/oic/res" Resource shall expose its "sduuid" and "sdname" Properties with values copied from the "uuid" and "name" Properties of the “/oic/sec/sdi” Resource, respectively.

If the "priv" Property is set to "true", then the "/oic/res" Resource shall not expose its "sduuid" and "sdname" Properties.

14 Security hardening guidelines/execution environment security

14.1 Preamble

This is an informative clause. Many TGs in OCF have security considerations for their protocols and environments. These security considerations are addressed through security mechanisms specified in the security documents for OCF. However, effectiveness of these mechanisms depends on security robustness of the underlying hardware and software Platform. This clause defines the components required for execution environment security.

14.2 Execution environment elements

14.2.1 Execution environment elements general

Execution environment within a computing Device has many components. To perform security functions in a robustness manner, each of these components has to be secured as a separate dimension. For instance, an execution environment performing AES cannot be considered secure if the input path entering keys into the execution engine is not secured, even though the partitions of the CPU, performing the AES encryption, operate in isolation from other processes. Different dimensions referred to as elements of the execution environment are listed below.

- (Secure) Storage
- (Secure) Execution engine
- (Trusted) Input/output paths
- (Secure) Time Source/clock
- (Random) number generator
- (Approved) cryptographic algorithms
- Hardware Tamper (protection)

NOTE Software security practices (such as those covered by Open Web Application Security Project) are outside scope of this document, as development of secure code is a practice to be followed by the open source development community. This document will however address the underlying Platform assistance required for executing software. Examples are secure boot and secure software upgrade.

Each of the elements above are described in the clauses 14.2.2, 14.2.3, 14.2.4, 14.2.5, 14.2.6, 14.2.7.

14.2.2 Secure storage

14.2.2.1 Secure storage general

Secure storage refers to the physical method of housing sensitive or confidential data ("Sensitive Data"). Such data could include but not be limited to symmetric or asymmetric private keys, certificate data, OCF Security Domain access credentials, or personal user information. Sensitive Data requires that its integrity be maintained, whereas Critical Sensitive Data requires that both its integrity and confidentiality be maintained.

It is strongly recommended that IoT Device makers provide reasonable protection for Sensitive Data so that it cannot be accessed by unauthorized Devices, groups or individuals for either malicious or benign purposes. In addition, since Sensitive Data is often used for authentication and encryption, it must maintain its integrity against intentional or accidental alteration.

A partial list of Sensitive Data is outlined in Table 58:

Table 58 – Examples of sensitive data

Data	Integrity protection	Confidentiality protection
Owner PSK (Symmetric Keys)	Yes	Yes
Service provisioning keys	Yes	Yes
Asymmetric Private Keys	Yes	Yes
Certificate Data and Signed Hashes	Yes	Not required
Public Keys	Yes	Not required
Access credentials (e.g. SSID, passwords, etc.)	Yes	Yes
ECDH/ECDH Dynamic Shared Key	Yes	Yes
Root CA Public Keys	Yes	Not required
Device and Platform IDs	Yes	Not required
Easy Setup Resources	Yes	Yes
Access Token	Yes	Yes

Exact method of protection for secure storage is implementation specific, but typically combinations of hardware and software methods are used.

14.2.2.2 Hardware secure storage

Hardware secure storage is recommended for use with critical Sensitive Data such as symmetric and asymmetric private keys, access credentials, and personal private data. Hardware secure storage most often involves semiconductor-based non-volatile memory ("NVRAM") and includes countermeasures for protecting against unauthorized access to Critical Sensitive Data.

Hardware-based secure storage not only stores Sensitive Data in NVRAM, but also provides protection mechanisms to prevent the retrieval of Sensitive Data through physical and/or electronic attacks. It is not necessary to prevent the attacks themselves, but an attempted attack should not result in an unauthorized entity successfully retrieving Sensitive Data.

Protection mechanisms should provide JIL Moderate protection against access to Sensitive Data from attacks that include but are not limited to:

1) Physical decapping of chip packages to optically read NVRAM contents
2) Physical probing of decapped chip packages to electronically read NVRAM contents
3) Probing of power lines or RF emissions to monitor voltage fluctuations to discern the bit patterns of Critical Sensitive Data
4) Use of malicious software or firmware to read memory contents at rest or in transit within a microcontroller
5) Injection of faults that induce improper Device operation or loss or alteration of Sensitive Data

14.2.2.3 Software storage

It is generally NOT recommended to rely solely on software and unsecured memory to store Sensitive Data even if it is encrypted. Critical Sensitive Data such as authentication and encryption keys should be housed in hardware secure storage whenever possible.

Sensitive Data stored in volatile and non-volatile memory shall be encrypted using acceptable algorithms to prevent access by unauthorized parties through methods described in 14.2.2.2.

14.2.2.4 Additional security guidelines and best practices

Some general practices that can help ensure that Sensitive Data is not compromised by various forms of security attacks:

1) FIPS Random Number Generator ("RNG") – Insufficient randomness or entropy in the RNG used for authentication challenges can substantially degrade security strength. For this reason, it is recommended that a FIPS 800-90A-compliant RNG with a certified noise source be used for all authentication challenges.

2) Secure download and boot – To prevent the loading and execution of malicious software, where it is practical, it is recommended that Secure Download and Secure Boot methods that authenticate a binary's source as well as its contents be used.

3) Deprecated algorithms – Algorithms included but not limited to the list below are considered unsecure and shall not be used for any security-related function:

   a) SHA-1

   b) MD5

   c) RC4

   d) RSA 1024

4) Encrypted transmission between blocks or components – Even if critical Sensitive Data is stored in Secure Storage, any use of that data that requires its transmission out of that Secure Storage should be encrypted to prevent eavesdropping by malicious software within an MCU/MPU.

5) It is recommended to avoid using wildcard in Subject Id ("*"), when setting up "/oic/sec/cred" Resource entries, since this opens up an identity spoofing opportunity.

6) Device vendor understands that it is the Device vendor's responsibility to ensure the Device meets security requirements for its intended uses. As an example, IoTivity is a reference implementation intended to be used as a basis for a product, but IoTivity has not undergone 3rd party security review, penetration testing, etc. Any Device based on IoTivity should undergo appropriate penetration testing and security review prior to sale or deployment.

7) Device vendor agrees to publish the expected support lifetime for the Device to OCF and to consumers. Changes should be made to a public and accessible website. Expectations should be clear as to what will be supported and for how long the Device vendor expects to support security updates to the software, operating system, drivers, networking, firmware and hardware of the device.

8) Device vendor has not implemented test or debug interfaces on the Device which are operable or which can be enabled which might present an attack vector on the Device which circumvents the interface-level security or access policies of the Device.

9) Device vendor understands that if an application running on the Device has access to cryptographic elements such as the private keys or Ownership Credential, then those elements have become vulnerable. If the Device vendor is implementing a Bridge, an OBT, or a Device with access to the Internet beyond the local network, the execution of critical functions should take place within a Trusted or Secure Execution Environment (TEE/SEE).

10) Any PINs or fixed passphrases used for onboarding, Wi-Fi Easy Setup, SoftAP management or access, or other security-critical function, should be sufficiently unique (do not duplicate passphrases. The creation of these passphrases or PINS should not be algorithmically deterministic nor should they use insufficient entropy in their creation.

11) Ensure that there are no remaining "VENDOR_TODO" items in the source code.

12) If the implementation of this document uses the "Just Works" onboarding method, understand that there is a man-in-the-middle vulnerability during the onboarding process where a malicious party could intercept messages between the device being onboarded and the OBT and could persist, acting as an intermediary with access to message traffic, during the lifetime of that

onboarded device. The recommended best practice would be to use an alternate ownership transfer method (OTM) instead of "Just Works".

13) It is recommended that at least one static and dynamic analysis tool[1] be applied to any proposed major production release of the software before its release, and any vulnerabilities resolved.

14.2.3 Secure execution engine

Execution engine is the part of computing Platform that processes security functions, such as cryptographic algorithms or security protocols (e.g. DTLS). Securing the execution engine requires the following

– Isolation of execution of sensitive processes from unauthorized parties/ processes. This includes isolation of CPU caches, and all of execution elements that needed to be considered as part of trusted (crypto) boundary.

– Isolation of data paths into and out of execution engine. For instance, both unencrypted but sensitive data prior to encryption or after decryption, or cryptographic keys used for cryptographic algorithms, such as decryption or signing. See clause 14.2.4 for more details.

14.2.4 Trusted input/output paths

Platform implementations should only expose information, network interfaces, ports and other functions that are necessary for the correct functioning of the Platform. It is also strongly recommended that Vendors configure a Platform to expose only a fixed set of explicitly documented open network ports and/or port ranges.

14.2.5 Secure clock

Many security functions depend on time-sensitive credentials. Examples are time stamped Kerberos tickets, OAUTH tokens, X.509 certificates, OSCP response, software upgrades, etc. Lack of secure source of clock can mean an attacker can modify the system clock and fool the validation mechanism. Thus an SEE needs to provide a secure source of time that is protected from tampering. Trustworthiness from security robustness standpoint is not the same as accuracy. Protocols such as NTP can provide rather accurate time sources from the network, but are not immune to attacks. A secure time source on the other hand can be off by seconds or minutes depending on the time-sensitivity of the corresponding security mechanism. Secure time source can be external as long as it is signed by a trusted source and the signature validation in the local Device is a trusted process (e.g. backed by secure boot).

14.2.6 Selecting cryptographic algorithms

When an implementation adds additional cryptographic algorithms on top of those define in this specification, then those shall be only publicly-vetted, peer-reviewed (e.g. NIST-approved) and non-deprecated.

14.2.7 Hardware tamper protection

Various levels of hardware tamper protection exist. We borrow FIPS 140-2 terminology (not requirements) regarding tamper protection for cryptographic module

– Production-grade (lowest level): this means components that include conformal sealing coating applied over the module’s circuitry to protect against environmental or other physical damage. This does not however require zeroization of secret material during physical maintenance. This definition is borrowed from FIPS 140-2 security level 1.

– Tamper evident/proof (mid-level), This means the Device shows evidence (through covers, enclosures, or seals) of an attempted physical tampering. This definition is borrowed from FIPS 140-2 security level 2.

[1] A general discussion of analysis tools can be found here: https://www.ibm.com/developerworks/library/se-static/

– Tamper resistance (highest level), this means there is a response to physical tempering that typically includes zeroization of sensitive material on the module. This definition is borrowed from FIPS 140-2 security level 3.

It is difficult of specify quantitative certification test cases for accreditation of these levels. Content protection regimes usually talk about different tools (widely available, specialized and professional tools) used to circumvent the hardware protections put in place by manufacturing. If needed, OCF can follow that model, if and when OCF engage in distributing sensitive key material (e.g. PKI) to its members.

14.3 Secure Boot

14.3.1 Concept of software module authentication

In order to ensure that all components of a Device are operating properly and have not been tampered with, it is best to ensure that the Device is booted properly. There may be multiple stages of boot. The end result is an application running on top an operating system that takes advantage of memory, CPU and peripherals through drivers.

The general concept is that each software module is invoked only after cryptographic integrity verification is complete. The integrity verification relies on the software module having been hashed (e.g. SHA_1, SHA_256) and then signed with a cryptographic signature algorithm with (e.g. RSA), with a key that only a signing authority has access to.

Figure 30 depicts software module authentication.

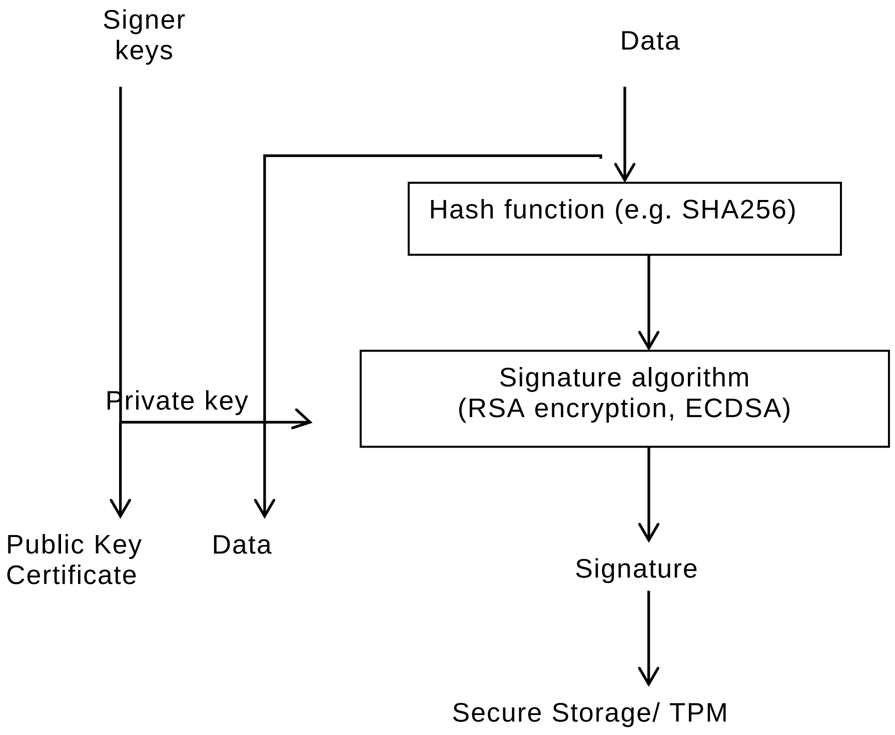


Figure 30 – Software module authentication

After the data is signed with the signer's signing key (a private key), the verification key (the public key corresponding to the private signing key) is provided for later verification. For lower level software modules, such as bootloaders, the signatures and verification keys are inserted inside tamper proof memory, such as one-time programmable memory or TPM. For higher level software modules, such as application software, the signing is typically performed according to the PKCS#7 format IETF RFC 2315, where the signed data format includes both indications for signature algorithm, hash algorithm as well as the signature verification key (or certificate). Secure boot does not require use of PKCS#7 format.

Figure 31 depicts verification software module.

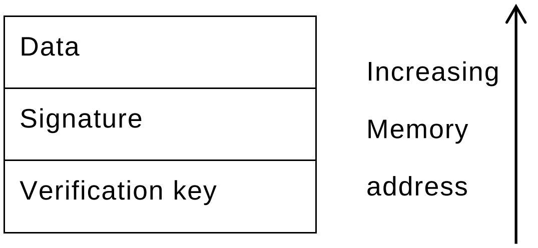


Figure 31 – Verification software module

As shown in Figure 32 the verification module first decrypts the signature with the verification key (public key of the signer). The verification module also calculates a hash of the data and then compares the decrypted signature (the original) with the hash of data (actual) and if the two values match, the software module is authentic.

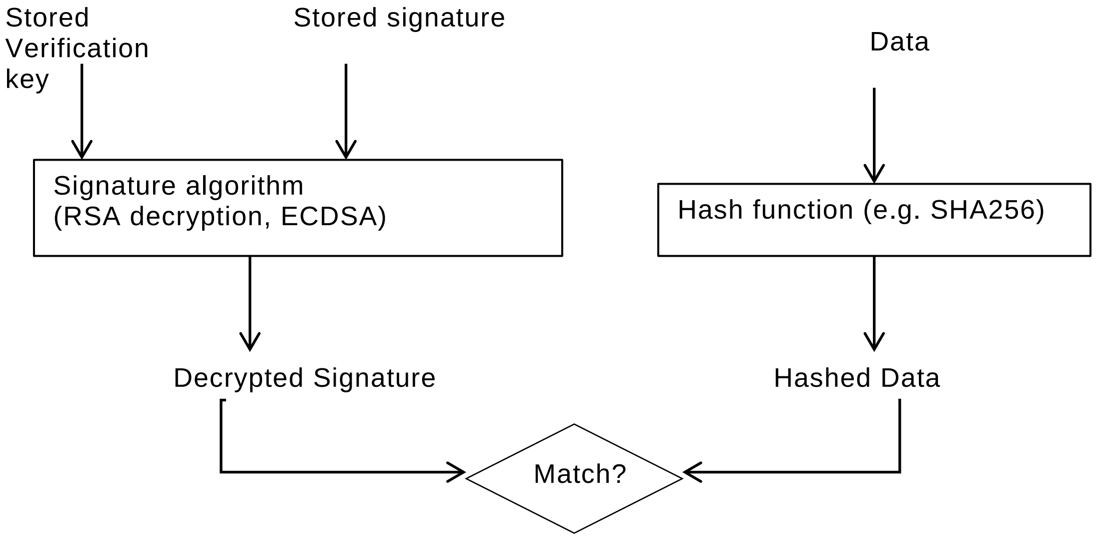


Figure 32 – Software module authenticity

14.3.2 Secure Boot process

Depending on the Device implementation, there may be several boot stages. Typically, in a PC/ Linux type environment, the first step is to find and run the BIOS code (first-stage bootloader) to find out where the boot code is and then run the boot code (second-stage boot loader). The second stage bootloader is typically the process that loads the operating system (Kernel) and transfers the execution to the where the Kernel code is. Once the Kernel starts, it may load external Kernel modules and drivers.

When performing a secure boot, it is required that the integrity of each boot loader is verified before executing the boot loader stage. As mentioned, while the signature and verification key for the lowest level bootloader is typically stored in tamper-proof memory, the signature and verification key for higher levels should be embedded (but attached in an easily accessible manner) in the data structures software.

14.3.3 Robustness requirements

14.3.3.1 Robustness general

To qualify as high robustness secure boot process, the signature and hash algorithms shall be one of the approved algorithms, the signature values and the keys used for verification shall be stored in secure storage and the algorithms shall run inside a secure execution environment and the keys shall be provided the SEE over trusted path.

14.3.3.2 Next steps

Develop a list of approved algorithms and data formats

14.4 Attestation

14.5 Software Update

14.5.1 Overview

The Device lifecycle does not end at the point when a Device is shipped from the manufacturer; the distribution, retailing, purchase, installation/onboarding, regular operation, maintenance and end-of-life stages for the Device remain outstanding. It is possible for the Device to require update during any of these stages, although the most likely times are during onboarding, regular operation and maintenance. The manufacturer shall have a defined policy available to OCF Security Domain Owner (e.g. via a website link) covering handling of any device vulnerabilities, including the software update information (e.g. if and how such updates are provided). This policy shall also cover any post end-of-life or end-of-service vulnerabilities. The aspects of the software include, but are not limited to, firmware, operating system, networking stack, application code, drivers, etc.

14.5.2 Recognition of current differences

Different manufacturers approach software update utilizing a collection of tools and strategies: over-the-air or wired USB connections, full or partial replacement of existing software, signed and verified code, attestation of the delivery package, verification of the source of the code, package structures for the software, etc.

It is recommended that manufacturers review their processes and technologies for compliance with industry best-practices that a thorough security review of these takes place and that periodic review continue after the initial architecture has been established.

This document applies to software updates as recommended to be implemented by OCF Devices; it does not have any bearing on the above-mentioned alternative proprietary software update mechanisms. The described steps are being triggered by an OCF Client, the actual implementation of the steps and how the software package is downloaded and upgraded is vendor specific.

The triggers that can be invoked from OCF clients can:

1) Check if new software is available
2) Download and verify the integrity of the software package
3) Install the verified software package

The triggers are not sequenced; each trigger can be invoked individually.

The state of the transitions of software update is in Figure 33.

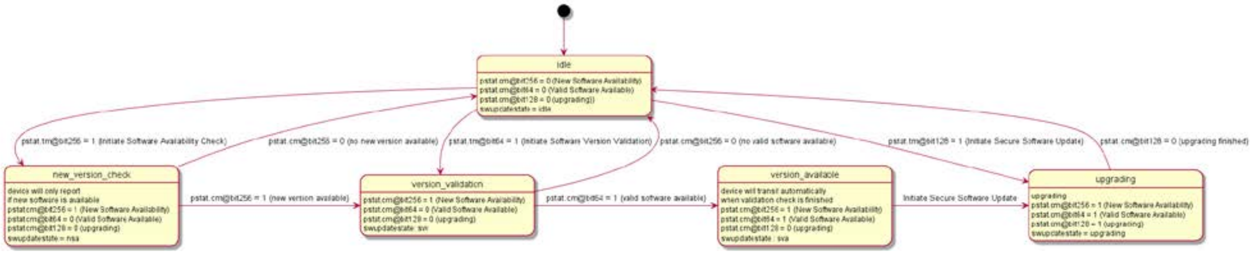

idle
pstat.cm@bit256 = 0 (New Software Availability)
pstat.cm@bit64 = 0 (Valid Software Available)
pstat.cm@bit128 = 0 (upgrading))
swupdatestate = idle
pstat.tm@bit256 = 1 (Initiate Software Availability Check)
pstat.cm@bit256 = 0 (no new version available)
pstat.tm@bit64 = 1 (Initiate Software Version Validation)
pstat.cm@bit256 = 0 (no valid software available)
pstat.tm@bit128 = 1 (Initiate Secure Software Update)
pstat.cm@bit128 = 0 (upgrading finished)
new_version_check
device will only report
if new software is available
pstatcm@bit256 = 1 (New Software Availability)
pstatcm@bit64 = 0 (Valid Software Available)
pstatcm@bit128 = 0 (upgrading)
swupdatestate = nsa
pstat.cm@bit256 = 1 (new version available)
version_validation
pstat.cm@bit256 = 1 (New Software Availability)
pstat.cm@bit64 = 0 (Valid Software Available)
pstat.cm@bit128 = 0 (upgrading)
swupdatestate: svv
pstat.cm@bit64 = 1 (valid software available)
version_available
device will transit automatically
when validation check is finished
pstat.cm@bit256 = 1 (New Software Availabilit)
pstat.cm@bit64 = 1 (Valid Software Available)
pstat.cm@bit128 = 0 (upgrading)
swupdatestate : sva
Initiate Secure Software Update
upgrading
upgrading
pstat.cm@bit256 = 1 (New Software Availability)
pstat.cm@bit64 = 1 (Valid Software Available)
pstat.cm@bit128 = 1 (upgrading)
swupdatestate = upgrading

Figure 33 – State transitioning diagram for software download

Table 59 – Description of the software update bits

Bit	TM property	CM property
Bit 9	Initiate Software Availability Check	New Software Available
Bit 7	Initiate Software Version Validation	Valid Software Available
Bit 8	Initiate Secure Software Update	Upgrading

14.5.2.1 Checking availability of new software

Setting the Initiate Software Availability Check bit in the "/oic/sec/pstat.tm" Property (see Table 39 of clause 13.8) indicates a request to initiate the process to check if new software is available, e.g. the process whereby the Device checks if a newer software version is available on the external endpoint. Once the Device has determined if an newer software version is available, it sets the Initiate Software Availability Check bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE), indicating that new software is available or to 0 (FALSE) if no newer software version is available, See also Table 59 where the bits in property TM indicates that the action is initiated and the CM bits are indicating the result of the action. The Device receiving this trigger is not downloading and not validating the software to determine if new software is available. The version check is determined by the current software version and the software version on the external endpoint. The determination if a software package is newer is vendor defined.

14.5.3 Software Version Validation

Setting the Initiate Software Version Validation bit in the "/oic/sec/pstat.tm" Property (see Table 39 of 13.8) indicates a request to initiate the software version validation process, the process whereby the Device validates the software (including firmware, operating system, Device drivers, networking stack, etc.) against a trusted source to see if, at the conclusion of the check, the software update process will need to be triggered (see clause 14.5.4). When the Initiate Software Version Validation bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged Client, the Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and initiates a software version check. Once the Device has determined if a valid software is available, it sets the Initiate Software Version Validation bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE) if an update is available or 0 (FALSE) if no update is available. To signal completion of the Software Version Validation process, the Device sets the Initiate Software Version Validation bit in the "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Software Version Validation bit of "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the validation process.The Software Version Validation process can download the software from the external endpoint to verify the integrity of the software package.

14.5.4 Software Update

The software of a Device shall be updatable.

Setting the Initiate Secure Software Update bit in the "/oic/sec/pstat.tm" Property (see Table 39 of clause 13.8) indicates a request to initiate the software update process. When the Initiate Secure Software Update bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged Client, the Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and initiates a software update process. Once the Device has completed the software update process, it sets the Initiate Secure Software Update bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE) if/when the software was successfully updated or 0 (FALSE) if no update was performed. To signal completion of the Secure Software Update process, the Device sets the Initiate Secure Software Update bit in the "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Secure Software Update bit of "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the update process.

14.5.4.1 State of Device after software update

The state of all Resources implemented in the Device should be the same as after boot, meaning that the software update is not resetting user data and retaining a correct state.

User data of a Device is defined as:

- Retain the SVR states, e.g. the on boarded state, registered clients.
- Retain all created Resources
- Retain all stored data of a Resource
  - For example the preferences stored for the brewing Resource ("oic.r.brewing").

14.5.5 Recommended usage

The Initiate Secure Software Update bit of "/oic/sec/pstat.tm" should only be set by a Client after the Initiate Software Version Validation check is complete.

The process of updating Device software may involve state changes that affect the Device Operational State ("/oic/sec/pstat.dos"). Devices with an interest in the Device(s) being updated should monitor "/oic/sec/pstat.dos" and be prepared for pending software update(s) to affect Device state(s) prior to completion of the update.

The Device itself may indicate that it is autonomously initiating a software version check/update or that a check/update is complete by setting the "pstat.tm" and "pstat.cm" Initiate Software Version Validation and Secure Software Update bits when starting or completing the version check or update process. As is the case with a Client-initiated update, Clients can be notified that an autonomous version check or software update is pending and/or complete by observing pstat Resource changes.

The "oic.r.softwareupdate" Resource Type specifies additional features to control the software update process see core specification.

14.6 Non-OCF Endpoint interoperability

14.7 Security levels

Security Levels are a way to differentiate Devices based on their security criteria. This need for differentiation is based on the requirements from different verticals such as industrial and health care and may extend into smart home. This differentiation is distinct from Device classification (e.g. IETF RFC 7228)

These categories of security differentiation may include, but is not limited to:

1) Security Hardening
2) Identity Attestation
3) Certificate/Trust
4) Onboarding Technique
5) Regulatory Compliance

    a) Data at rest
    b) Data in transit
6) Cipher Suites – Crypto Algorithms & Curves
7) Key Length
8) Secure Boot/Update

In the future security levels can be used to define interoperability.

The following applies to the OCF Security Specification 1.1:

The current document does not define any other level beyond Security Level 0. All Devices will be designated as Level 0. Future versions may define additional levels.

Additional comments:

- The definition of a given security level will remain unchanged between versions of the document.
- Devices that meet a given level may, or may not, be capable of upgrading to a higher level.
- Devices may be evaluated and re-classified at a higher level if it meets the requirements of the higher level (e.g. if a Device is manufactured under the 1.1 version of the document, and a later document version defines a security level 1, the Device could be evaluated and classified as level 1 if it meets level 1 requirements).
- The security levels may need to be visible to the End User.

14.8 Security Profiles

14.8.1 Security Profiles general

Security Profiles are a way to differentiate OCF Devices based on their security criteria. This need for differentiation is based on the requirements from different verticals such as industrial and health care and may extend into smart home. This differentiation is distinct from device classification (e.g. IETF RFC 7228)

These categories of security differentiation may include, but is not limited to:

1) Security Hardening and assurances criteria
2) Identity Attestation
3) Certificate/Trust
4) Onboarding Technique
5) Regulatory Compliance
    a) Data at rest
    b) Data in transit
6) Cipher Suites – Crypto Algorithms & Curves
7) Key Length
8) Secure Boot/Update

Each Security Profile definition shall specify the version or versions of the OCF Security Specification(s) that form a baseline set of normative requirements. The profile definition may include security requirements that supersede baseline requirements (not to relax security requirements).

Security Profiles have the following properties:

- A given profile definition is not specific to the version of the document that defines it. For example, the profile may remain constant for subsequent OCF Security Specification versions.

– A specific OCF Device and platform combination may be used to satisfy the security profile.

– Profiles may have overlapping criteria; hence it may be possible to satisfy multiple profiles simultaneously.

– An OCF Device that satisfied a profile initially may be re-evaluated at a later time and found to satisfy a different profile (e.g. if a device is manufactured under the 1.1 version of the document, and a later document version defines a security profile Black, the device could be evaluated and classified as profile Black if it meets profile Black requirements).

– A machine-readable representation of compliance results specifically describing profiles satisfied may be used to facilitate OCF Device onboarding. (e.g. a manufacturer certificate or manifest may contain security profiles attributes).

14.8.2 Identification of Security Profiles (Normative)

14.8.2.1 Security Profiles in prior documents

OCF Devices conforming to versions of the OCF Security Specifications where Security Profiles Resource was not defined may be presumed to satisfy the "sp-baseline-v0" profile (defined in 14.8.3.3) or may be regarded as unspecified. If Security Profile is unspecified, the Client may use the OCF Security Specification version to characterize expected security behaviour.

14.8.2.2 Security Profile Resource definition

The "/oic/sec/sp" Resource is used by the OCF Device to show which OCF Security Profiles the OCF Device is capable of supporting and which are authorized for use by the OCF Security Domain owner. Properties of the Resource identify which OCF Security Profile is currently operational. The ocfSecurityProfileOID value type shall represent OID values and may reference an entry in the form of strings (UTF-8).

"/oic/sec/sp" Resource is defined in Table 60.

Table 60 – Definition of the "/oic/sec/sp" Resource

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/sp	Security Profile Resource Definition	oic.r.sp	oic.if.baseline, oic.if.rw	Resource specifying supported and current security profile(s)	Discoverable

Table 61 defines the Properties of "/oic/sec/sp" Resource.

Table 61 – Properties of the "/oic/sec/sp" Resource

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Supported Security Profiles	supportedprofiles	ocfSecurityProfileOID	array	RW	Yes	Array of supported Security Profiles (e.g. ["1.3.6.1.4.1.51414.0.0.2.0","1.3.6.1.4.1.51414.0.0.3.0"])
SecurityProfile	currentprofile	ocfSecurityProfileOID	N/A	RW	Yes	Currently active Security Profile (e.g. "1.3.6.1.4.1.51414.0.0.3.0")

The following OIDs are defined to uniquely identify Security Profiles. Future Security Profiles or changes to existing Security Profiles may result in a new ocfSecurityProfileOID.

```
id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
                                      private(4) enterprise(1) OCF(51414) }

  id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }

    id-ocfSecurityProfile ::= { id-ocfSecurity 0 }
```

```
        sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
        --The Security Profile is not specified
        sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
        --This specifies the OCF Baseline Security Profile(s)
        sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
        --This specifies the OCF Black Security Profile(s)
        sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
        --This specified the OCF Blue Security Profile(s)
        sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }
        --This specifies the OCF Purple Security Profile(s)

        --versioned Security Profiles
        sp-unspecified-v0 ::= ocfSecurityProfileOID (id-sp-unspecified 0}
        --v0 of unspecified security profile, "1.3.6.1.4.1.51414.0.0.0.0"
        sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
        --v0 of baseline security profile, "1.3.6.1.4.1.51414.0.0.1.0"
        sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
        --v0 of black security profile, "1.3.6.1.4.1.51414.0.0.2.0"
        sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
        --v0 of blue security profile, "1.3.6.1.4.1.51414.0.0.3.0"
        sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
        --v0 of purple security profile, "1.3.6.1.4.1.51414.0.0.4.0"

        ocfSecurityProfileOID ::= UTF8String
```

14.8.3 Security Profiles

14.8.3.1 Security Profiles general

The Security Profiles Resource shall be pre-populated with manufacturer default values (Refer to the Security Profile clauses for additional details).

The OCF Conformance criteria may require vendor attestation that establishes the expected environment in which the OCF Device is hosted (Refer to the Security Profile clauses for specific requirements).

14.8.3.2 Security Profile Unspecified (sp-unspecified-v0)

The Security Profile "sp-unspecified-v0" is reserved for future use.

14.8.3.3 Security Profile Baseline v0 (sp-baseline-v0)

The Security Profile "sp-baseline-v0" is defined for all OCF Security Specification versions where the "/oic/sec/sp" Resource is defined. All Devices shall include the "sp-baseline-v0" OID in the "supportedprofiles" Property of the "/oic/sec/sp" Resource.

It indicates the OCF Device satisfies the normative security requirements for this document.

When a device supports the baseline profile, the "supportedprofiles" Property shall contain sp-baseline-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.1.0", and may contain other profiles.

When a manufacturer makes sp-baseline-v0 the default, by setting the "currentprofile" Property to "1.3.6.1.4.1.51414.0.0.1.0", the "supportedprofiles" Property shall contain sp-baseline-v0.

14.8.3.4 Security Profile Black (sp-black-v0)

14.8.3.4.1 Black Profile general

The need for Security Profile Black v0 is to support devices and manufacturers who wish to certify their devices meeting this specific set of security criteria. A Device may satisfy the Black requirements as well as requirements of other profiles, the Black Security Profile is not necessarily mutually exclusive with other Security Profiles unless those requirements conflict with the explicit requirements of the Black Security Profile.

14.8.3.4.2 Devices targeted for Security Profile Black v0

Security Profile Black devices could include any device a manufacturer wishes to certify at this profile, but healthcare devices and industrial devices with additional security requirements are the initial target. Additionally, manufacturers of devices at the edge of the network (or fog), or devices with exceptional profiles of trust bestowed upon them, may wish to certify at this profile; these types of devices may include, but are not limited to the following:

- Bridges (Mapping devices between ecosystems handling virtual devices from different ecosystems)
- Resource Directories (Devices trusted to manage OCF Security Domain Resources)
- Remote Access (Devices which have external access but can also act within the OCF Security Domain)
- Healthcare Devices (Devices with specific requirements for enhanced security and privacy)
- Industrial Devices (Devices with advanced management, security and attestation requirements)

14.8.3.4.3 Requirements for certification at Security Profile Black (normative)

Every device with "currentprofile" Property of the "/oic/sec/sp" Resource designating a Security Profile of "sp-black-v0", as defined in clause 14.8.2, shall support each of the following:

- Onboarding via OCF Rooted Certificate Chain, including PKI chain validation
- Support for AES 128 encryption for data at rest and in transit.
- Hardening minimums: manufacturer assertion of secure credential storage
- In enumerated item #10 “The "/oic/sec/cred" Resource should contain credential(s) if required by the selected OTM” is changed to require the credential be stored: “The "/oic/sec/cred" Resource shall contain credential(s).”
- The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its certificate and the extension's 'securityProfile' field shall contain sp-black-v0 represented by the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.2.0".

When a device supports the black profile, the "supportedprofiles" Property shall contain sp-black-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.2.0", and may contain other profiles.

When a manufacturer makes sp-black-v0 the default, by setting the "currentprofile" Property to "1.3.6.1.4.1.51414.0.0.2.0", the "supportedprofiles" Property shall contain sp-black-v0.

The OCF Rooted Certificate Chain and PKI Is defined by and structured within a framework described in the supporting documents:

- Certificate Profile (See 9.4.2)
- Certificate Policy (see Certificate Policy document: https://openconnectivity.org/specs/OCF%20Certificate%20Policy.pdf)

14.8.3.5 Security Profile Blue v0 (sp-blue-v0)

14.8.3.5.1 Blue Profile general

The Security Profile Blue is used when manufacturers issue platform certificates for platforms containing manufacturer-embedded keys. Compatibility with interoperable trusted platforms is anticipated using certificate extensions defined by the Trusted Computing Group (TCG). OCF Security Domain owners evaluate manufacturer supplied certificates and attributed data to determine an appropriate OCF Security Profile that is configured for OCF Devices at onboarding. OCF Devices may satisfy multiple OCF Security Profiles. The OCF Security Domain owner may configure deployments using the Security Profile as OCF Security Domain partitioning criteria.

Certificates issued to Blue Profile Devices shall be issued by a CA conforming to the CA Vetting Criteria defined by OCF.

14.8.3.5.2 Platforms and Devices for Security Profile Blue v0

The OCF Security Profile Blue anticipates an ecosystem where platform vendors may differ from OCF Device vendor and where platform vendors may implement trusted platforms that may conform to industry standards defining trusted platforms. The OCF Security Profile Blue specifies mechanisms for linking platforms with OCF Device(s) and for referencing quality assurance criteria produced by OCF conformance operations. The OCF Security Domain owner evaluates these data when an OCF Device is onboarded into the OCF Security Domain. Based on this evaluation the OCF Security Domain owner determines which Security Profile may be applied during OCF Device operation. All OCF Device types may be considered for evaluation using the OCF Security Profile Blue.

14.8.3.5.3 Requirements for certification at Security Profile Blue v0

The OCF Device satisfies the Blue profile v0 (sp-blue-v0) when all of the security normative for this document version are satisfied and the following additional criteria are satisfied.

OCF Blue profile defines the following OCF Device quality assurances:

- The OCF Conformance criteria shall require vendor attestation that the conformant OCF Device was hosted on one or more platforms that satisfies OCF Blue platform security assurances and platform security and privacy functionality requirements.
- The OCF Device achieving OCF Blue Security Profile compliance will be registered by OCF and published by OCF in a machine readable format.
- The OCF Blue Security Profile compliance registry may be digitally signed by an OCF owned signing key.
- The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its certificate and the extension's 'securityProfile' field shall contain sp-blue-v0 represented by the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.3.0".
- The OCF Device shall include an X.509v3 OCF CPL Attributes Extension (clause 9.4.2.2.7) in its certificate.
- The DOTS is expected to perform a lookup of the certification status of the OCF Device using the OCF CPL Attributes Extension values and verify that the sp-blue-v0 OID is listed in the extension's "securityprofiles" field.

OCF Blue profile defines the following OCF Device security functionality:

- OCF Device(s) shall be hosted on a platform where a cryptographic and secure storage functions are hardened by the platform.
- OCF Device(s) hosted on a platform shall expose accompanying manufacturer credentials using the "/oic/sec/cred" Resource where the "credusage" Property contains the value "oic.sec.cred.mfgcert".
- OCF Device(s) that are hosted on a TCG-defined trusted platform should use an IEEE802.1AR IDevID and should verify the "TCG Endorsement Key Credential". All TCG-defined manufacturer credentials may be identified by the "oic.sec.cred.mfgcert" value of the "credusage" Property of the "/oic/sec/cred" Resource. They may be used in response to selection of the "oic.sec.doxm.mfgcert" owner transfer method.
- OCF Device(s) shall use AES128 equivalent minimum protection for transmitted data. (See NIST SP 800-57).
- OCF Device(s) shall use AES128 equivalent minimum protection for stored data. (See NIST SP 800-57).

- OCF Device(s) should use AES256 equivalent minimum protection for stored data. (See NIST SP 800-57).
- OCF Device(s) should protect the "/oic/sec/cred" Resource using the platform provided secure storage.
- OCF Device(s) shall protect trust anchors (aka policy defining trusted CAs and pinned certificates) using platform provided secure storage.
- OCF Device(s) should check certificate revocation status for locally issued certificates.
- The DOTS is expected to check certificate revocation status for all certificates in manufacturer certificate path(s) if available. If a certificate is revoked, certificate validation fails and the connection is refused. The DOTS may disregard revocation status results if unavailable.

OCF Blue profile defines the following platform security assurances:

- Platforms implementing cryptographic service provider (CSP) functionality and secure storage functionality should be evaluated with a minimum FIPS140-2 Level 2 or Common Criteria EAL Level 2.
- Platforms implementing trusted platform functionality should be evaluated with a minimum Common Criteria EAL Level 1.

OCF Blue profile defines the following platform security and privacy functionality:

- The Platform shall implement cryptographic service provider (CSP) functionality.
- Platform CSP functionality shall include cryptographic algorithms, random number generation, secure time.
- The Platform shall implement AES128 equivalent protection for transmitted data. (See NIST SP 800-57).
- The Platform shall implement AES128 and AES256 equivalent protection for stored data. (See NIST SP 800-57).
- Platforms hosting OCF Device(s) should implement a platform identifier following IEEE802.1AR or Trusted Computing Group(TCG) specifications.
- Platforms based on Trusted Computing Group (TCG) platform definition that host OCF Device(s) should supply TCG-defined manufacture certificates; also known as "TCG Endorsement Key Credential" (which complies with IETF RFC 5280) and "TCG Platform Credential" (which complies with IETF RFC 5755).

When a device supports the blue profile, the "supportedprofiles" Property shall contain sp-blue-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.3.0", and may contain other profiles.

When a manufacturer makes sp-blue-v0 the default, by setting the "currentprofile" Property to "1.3.6.1.4.1.51414.0.0.3.0", the "supportedprofiles" Property shall contain sp-blue-v0.

During onboarding, while the device state is RFOTM, the DOTS may update the "currentprofile" Property to one of the other values found in the "supportedprofiles" Property.

14.8.3.6 Security Profile Purple v0 (sp-purple-v0)

Every device with the "/oic/sec/sp" Resource designating "sp-purple-v0", as defined in clause 14.8.2 shall support following minimum requirements

- Hardening minimums: secure credential storage, software integrity validation, secure update.
- If a Certificate is used, the OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its certificate and the extension's 'securityProfile' field shall contain sp-purple-v0 represented by the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.4.0"
- The OCF Device shall include a X.509v3 OCFCPLAttributes Extension (clause 9.4.2.2.7) in its End-Entity Certificate when manufacturer certificate is used.

Security Profile Purple has following optional security hardening requirements that the device may additionally support.

– Hardening additions: secure boot, hardware backed secure storage

– The OCF Device shall include a X.509v3 OCFSecurityClaims Extension (clause 9.4.2.2.6) in its End-Entity Certificate and it shall include corresponding OIDs to the hardening additions implemented and attested by the vendor. If there is no additional support for hardening requirements, X.509v3 OCFSecurityClaims Extension shall be omitted.

For software integrity validation, OCF Device(s) shall provide the integrity validation mechanism for security critical executables such as cryptographic modules or secure service applications, and they should be validated before the execution. The key used for validating the integrity should be explicitly trusted by the validating software module and stored outside of the software to be updated.

For secure update, OCF Device(s) shall be able to update its firmware in a secure manner.

For secure boot, OCF Device(s) shall implement the BIOS code (first-stage bootloader on ROM) to be executed by the processor on power-on, and secure boot parameters to be provisioned by tamper-proof memory. Also OCF Device(s) shall provide software module authentication for the security critical executables and stop the boot process if any integrity of them is compromised.

For hardware backed secure storage, OCF Device(s) shall store sensitive data in non-volatile memory ("NVRAM") and prevent the retrieval of sensitive data through physical and/or electronic attacks.

More details on security hardening guidelines for software integrity validation, secure boot, secure update, and hardware backed secure storage are described in 14.3, 14.5 and 14.2.2.2.

Certificates issued to Purple Profile Devices shall be issued by a CA conforming to the CA Vetting Criteria defined by OCF.

When a device supports the purple profile, the "supportedprofiles" Property shall contain sp-purple-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.4.0", and may contain other profiles.

When a manufacturer makes sp-purple-v0 the default, by setting the "currentprofile" Property to "1.3.6.1.4.1.51414.0.0.4.0", the "supportedprofiles" Property shall contain sp-purple-v0.

15 Device Type Specific requirements

15.1 Bridging security

15.1.1 Universal requirements for Bridging to another Ecosystem

The Bridge shall go through OCF ownership transfer as any other onboardee would.

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

Each VOD shall be onboarded by an OCF OBT. Each Virtual Bridged Device should be provisioned as appropriate in the Bridged Protocol. In other words, VODs and Virtual Bridged Devices are treated the same way as physical Devices. They are entities that have to be provisioned in their network.

Each VOD shall implement the behaviour required by ISO/IEC 30118-1 and this document. Each VOD shall perform authentication, access control, and encryption according to the security settings it received from the OCF OBT. Each Virtual Bridged Device shall implement the security requirements of the Bridged Protocol.

In addition, in order to be considered secure from an OCF perspective, the Bridge Platform shall use appropriate ecosystem-specific security options for communication between the Virtual Bridged Devices instantiated by the Bridge and Bridged Devices. This security shall include mutual authentication, and encryption and integrity protection of messages in the bridged ecosystem.

A VOD may authenticate itself to the DOTS using the Manufacturer Certificate Based OTM (see clause 7.3.6) with the Manufacturer Certificate and corresponding private key of the Bridge which instantiated that VOD.

A VOD may authenticate itself to the OCF Cloud using the Manufacturer Certificate and corresponding private key of the Bridge which instantiated that VOD.

A Bridge and the VODs created by that Bridge shall operate as independent Devices, with the following exceptions:

- If a Bridge creates a VOD while the Bridge is in an Unowned State, then the VOD shall be created in an Unowned State.
- An Unowned VOD shall not accept DTLS connection attempts nor TLS connection attempts nor any other requests, including discovery requests, while the Bridge (that created that VOD) is Unowned.
- At any time when a Bridge is transitioning from Owned to Unowned State, all Unowned VODs (created by that Bridge prior to the transition) shall drop any existing TLS and/or DTLS connections.
- At any time when a Bridge is transitioning from Unowned to Owned State, the Bridge shall trigger all Unowned VODs (created by that Bridge prior to the transition) to become accessible in RFOTM, with internal state as if the VOD has just transitioned from RESET to RFOTM.
- If a Bridge creates a VOD while the Bridge is in an Owned State, then the VOD shall become accessible in RFOTM, with internal state as if the VOD has just transitioned from RESET to RFOTM.

Table 62 intends to clarify this behaviour.

Table 62 – Dependencies of VOD Behaviour on Bridge state, as clarification of accompanying text

Bridge state	Additional dependencies on VOD behaviour	
	VOD is Unowned (either just created, or created previously)	**VOD is Owned**
From unboxing Bridge until just prior to the end of transition of Bridge from Unowned to Owned	No accepting DTLS connection attempts nor TLS connection attempts nor any other requests, including discovery requests	Not applicable
At end of transition from Unowned to Owned	VOD becomes accessible in RFOTM following Bridge's transition. Internal state as if just transitioned from RESET.	As per normal Device
Owned	As per normal Device	As per normal Device
At Start of transition from Owned to Unowned	Drop any established TLS/DTLS connections, even if already partway through Device ownership	As per normal Device
Start of transition from Owned to Unowned, until just prior to the end of transition from Unowned to Owned.	No accepting DTLS connection attempts nor TLS connection attempts nor any other requests, including discovery requests	As per normal Device

The "vods" Property of the "oic.r.vodlist" Resource on a Bridge reflects the details of all currently Owned VODs which have been created by that Bridge since the most recent hardware reset (if any) of the Bridge Platform (which removes all the created VODs), regardless of whether the VODs have the same owner as the Bridge or not. The entries in the "vods" Property are added and removed according to the following criteria:

– Whenever a VOD created by a Bridge transitions from being Unowned to being Owned, then an entry for that VOD shall be added to the "vods" Property of the "oic.r.vodlist" Resource of that Bridge.

– Whenever a VOD created by a Bridge transitions from being Owned to being Unowned, then entry for that VOD shall be removed from the "vods" Property of the "oic.r.vodlist" Resource of that Bridge. If that Bridge is currently in Unowned state, then the "oic.r.vodlist" Resource is not accessible, and the entry for that VOD shall be removed from the "vods" Property before or during the transition of that Bridge to the Owned state.

– All other modifications of the list are not allowed.

A Bridge shall only expose a secure OCF Endpoint for the "oic.r.vodlist" Resource.

15.1.2 Additional security requirements specific to Bridged protocols

15.1.2.1 Additional security Requirements specific to the AllJoyn protocol

For AllJoyn translator, an authenticated and authorized Client 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 specified in ISO/IEC 30118-3

15.1.2.2 Additional security requirements specific to the Bluetooth LE protocol

A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don’t communicate securely with the Bridge.

15.1.2.3 Additional security requirements specific to the oneM2M protocols

The Bridge shall implement oneM2M application access control as defined in the oneM2M Release 3 Specifications.

An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don’t communicate securely with the Bridge.

15.1.2.4 Additional security requirements specific to the U+ protocol

A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don’t communicate securely with the Bridge.

15.1.2.5 Additional security requirements specific to the Z-Wave protocol

A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don’t communicate securely with the Bridge.

15.1.2.6 Additional security requirements specific to the Zigbee protocol

A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don’t communicate securely with the Bridge.

15.1.2.7 Additional security requirements specific to the EnOcean Radio protocol

A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don’t communicate securely with the Bridge.

.

16 Alternative in-transit protection mechanisms

16.1 Introduction to in-transit protection mechanisms

In addition to the DTLS protection mechanisms for device-to-device communication specified in clause 9.4.7 and clause 11.2, and TLS protection specified in OCF Cloud Security Specification, OCF supports the following in-transit protection mechanisms:

- End-to-End Security of Unicast Messages using OSCORE, specified in clause 16.2.
- Simple Secure Multicast, specified in clause 16.3

16.2 End-to-End Security of Unicast Messages using OSCORE

16.2.1 Introduction to End-to-End Security of Unicast Messages using OSCORE

End-to-End Security of Unicast Messages is accomplished by applying a layer of in-transit protection above the transport layer Security (provided by DTLS or TLS) and below the resource-access authorization layer, using Object Security for Constrained RESTful Environments (OSCORE) IETF RFC 8613.

Relative to an exchange of an OCF CRUDN Request message and OCF CRUDN Response message:

- The Device acting as a Client (that is, sending an OCF CRUDN Request message and receiving the corresponding OCF CRUDN Response message) acts as an OSCORE client. Within the scope of clause 16.2, all Clients are assumed to support OSCORE and perform OSCORE client processing.
- The Device acting as a Server (that is, receiving an OCF CRUDN Request message and sending one or more corresponding OCF CRUDN Response messages) acts as an OSCORE server. Within the scope of clause 16.2, all Servers are assumed to support OSCORE and perform OSCORE server processing.

Clause 16.2.4 specifies the supported mechanism for establishing an OSCORE Security Context between two Devices. For each Device, an authorized Client (e.g. OBT) provisions the OSCORE Security Context parameters to a credential entry of the "/oic/sec/cred" Resource. The "subjectuuid" of that credential entry identifies the other Device that shares that OSCORE Security Context (similar to how a DTLS endpoint associates each DTLS PSK session with the Device UUID of the other DTLS endpoint).

16.2.2 OSCORE ID Namespace Prefix

Clause 16.2.4 specifies one mechanism for establishing an OSCORE Security Context between two Devices. Different mechanisms have different entities responsible for managing the selection of OSCORE Sender ID and OSCORE Recipient ID. There is value in preventing Devices having multiple OSCORE Security Contexts with identical Recipient IDs: this simplifies processing and avoids inefficiencies.

If a set of one or more coordinated entities (e.g. a group of OBTs) assigns a set of OSCORE Recipient IDs to OSCORE Security Contexts on a Device, then that set of entities is responsible for avoiding duplicate OSCORE Recipient IDs. However, two non-coordinated entities assigning OSCORE Recipient IDs might assign identical OSCORE Recipient IDs if there is no predefined agreement on assignment of OSCORE Recipient IDs.

For this reason, the first byte of the OSCORE Sender ID and OSCORE Recipient ID use a OSCORE Identifier Namespace Prefix. The Table Y is the authoritative definition of the assigned OSCORE Identifier Namespace Prefix values.

Table 63 – OSCORE Identifier Namespace Prefix

Value	Interpretation	Applicable clauses
0x00	Reserved for future use	
0x01	Directly provisioned OSCORE Security Context	16.2.4
0x02	Simple Secure Multicast	16.3
0x03-0x0F	Reserved for future use	

16.2.3 OSCORE protection and verification of unicast OCF CRUDN messages

All OSCORE message processing requirements in clause 8 in IETF RFC 8613 apply.

NOTE 1: Clause 8 in IETF RFC 8613 requires the Client keep the association of the request Token (see IETF RFC 7252) with the Security Context and Partial IV of the request, in order to be able to find the Security Context and compute the OSCORE Additional Authenticated Data when verifying the response.

If a Client has an established OSCORE Security Context associated with a Server, then the following call flow applies whenever the Client sends unicast OCF CRUDN request targeting Resources hosted on the Server. The Client may send multiple OSCORE requests to multiple Servers

1) The Client shall apply the OSCORE request protection processing to OCF CRUDN requests targeting Resources hosted on the Server as specified in clause 8.1 in IETF RFC 8613, using the OSCORE Security Context. See ISO/IEC 30118-1 for details on setting the Proxy-URI option.
   The Client sends the OSCORE request message to the Server (optionally via OCF Proxies). The OSCORE request message shall be delivered over secure transports: Device-to-Device communication is secured as specified in clause 9.4.7; Device to Cloud communication is secured as specified in OCF Cloud Specification and OCF Cloud Security Specification; and Cloud-to-Cloud communication is secured as specified in OCF Cloud API for Cloud Services Specification.
2) The Server receives a unicast OSCORE request message. The Server shall apply the OSCORE request verification and decryption processing in clause 8.2 of IETF RFC 8613 with the following clarifications:
   a) At Step 2 in clause 8.2 of IETF RFC 8613
      i) If either the decompression or the COSE message fails to decode, the Server shall respond with error response message (e.g. "Bad Option") including an Outer Max-Age option with value zero.
      ii) The Server attempts to retrieve the OSCORE Security Contexts associated with the Recipient ID in the ’kid’ parameter. If the Server fails to retrieve a OSCORE Security Context with OSCORE Recipient ID corresponding to the 'kid' parameter received, then the Server shall respond with an error response message (e.g. "Unauthorized") including an Outer Max-Age option with value zero.
   b) At step 6 in clause 8.2 of IETF RFC 8613, if the decryption failed then the Server shall respond with an error response message (e.g. "Bad Request) including an Outer Max-Age option with value zero.
   c) If a Server exposes one or more observable Resources, then the Server shall support receiving OSCORE request messages using the Observe option.
3) The Server shall process the OCF CRUDN request message (encapsulated in the OSCORE request message) resulting in OCF CRUDN response message(s). The Server shall treat the value of "subjectuuid" in the credential entry which contains the OSCORE Security Context used to verify and decrypt the OSCORE request message in Step 2 as Client's Device UUID for access control processing. The Server shall treat the connection type as "auth-crypt" for access control processing.

NOTE 2: Multiple OCF CRUDN response messages are only sent in scenarios where the OCF CRUDN Request message is an Observe Request message.

4) The Server shall apply the OSCORE response protection processing of clause 8.3 of IETF RFC 8613 to each OCF CRUDN response message, using the OSCORE Security Context used to successfully decrypt the OSCORE request (in Step 2 of the present clause). At Step 3 in clause 8.3 of IETF RFC 8613, the Server shall compute the AEAD nonce as described in clause 5.2 of IETF RFC 8613 by applying the following steps:

   a) Encode the Partial IV (OSCORE Sender Sequence Number in network byte order) and increment the OSCORE Sender Sequence Number by one.

   b) Compute the OSCORE AEAD nonce from the Sender ID, Common IV, and Partial IV.

   The Server shall support sending the OCF CRUDN response messages using the Observe option in OSCORE response messages. If an OCF CRUDN response message uses the Observe option, then the OSCORE response message shall include an Outer Max-Age option with value zero. The Server sends the OSCORE response message to the Client (optionally via OCF Proxies). As with the OSCORE request message, the OSCORE response message shall be delivered over secure transports - see Step 1 for details.

   The Server shall update the value of the "ssn" Property in the matching credential entry of the "/oic/sec/cred" Resource to reflect the next value of the OSCORE Sender Sequence Number to be sent to a corresponding Endpoint.

NOTE 3: If a Client retrieves the "/oic/sec/cred" Resource over the OSCORE channel, the OSCORE Sender Sequence Number in the header of the OSCORE message is expected to match the "ssn" value within the Resource representation.

5) The Client receives the OSCORE response message. The Client uses the Token (see IETF RFC 7252) in this response message to determine the corresponding OCF CRUDN request message, the OSCORE Security Context and Partial IV in Step 1 of the present clause; see Note 1. The Client shall apply OSCORE response protection processing of clause 8.3 of IETF RFC 8613 using this OSCORE Security Context and Partial IV. The Client should ignore a success response to an OSCORE-protected request if the response is not an OSCORE response message (indicated by the presence of the OSCORE option).

16.2.4 Direct provisioning of an OSCORE Security Context

This is a mechanism for establishing an OSCORE Security Context for communication between two Endpoints. All configurable parameters of the OSCORE Security Context are either:

- fixed to the OSCORE-specified default value, or
- directly provisioned by an authorized Client (e.g. OBT) to a credential entry of the "/oic/sec/cred" Resource of the two Endpoints.

The following OSCORE Security Context parameters shall use the default values defined in clause 3.2 of IETF RFC 8613 (this information is not configured by the OBT):

- AEAD Algorithm,
- HKDF,
- Replay Window,
- Master Salt,
- ID Context.

The following OSCORE Security Context parameters and associated Device UUID shall be provisioned to a credential entry of "/oic/sec/cred" of the Device:

- The "subjectuuid" shall be set to the deviceUUID of the other Endpoint to be associated with the OSCORE Security Context.
- The "credtype" shall be set to the value specified for a directly provisioned OSCORE Security Context in Table 22, clause 13.3.1.

- The "privatedata" Property of the credential entry shall be set to the 256-bit secret generated by the provisioning client (e.g. OBT). This value shall be used as the OSCORE Master Secret. Two Endpoints provisioned using this mechanism can communicate securely only if provisioned with identical values for the OSCORE Master Secret.
- The OSCORE Configuration parameters ("oscore") Property shall be present, and shall include the following Properties:
  - The OSCORE Sender ID of the OSCORE Security Context is in the "senderid" Property. That value shall be set to the hexadecimal representation of a 56-bit value selected by the provisioning Client (e.g. OBT). When using the mechanism described in the present clause, the first byte of this value is expected to have the value assigned in Table 63 for a directly provisioned OSCORE Security Context.
  - The OSCORE Recipient ID of the OSCORE Security Context is in the "recipientid" Property. That value shall be set to the hexadecimal representation of a 56-bit value selected by the provisioning Client (e.g. OBT). The first byte of this value is expected to have the value assigned in Table 63 for a directly provisioned OSCORE Security Context.

NOTE: The values for the OSCORE Sender ID and OSCORE Recipient ID of the OSCORE Security Context for one Device are provisioned as the values for the OSCORE Recipient ID and OSCORE Sender ID of the OSCORE Security Context for the other Device respectively.

On Device powering down, for each such credential entry, the Device shall write the value of corresponding OSCORE Sender Sequence Number as "ssn" Property to non-volatile memory. In event of a crash, devices should apply Appendix B.1.1 of IETF RFC 8613.

16.3 Simple Secure Multicast

16.3.1 Introduction to Simple Secure Multicast

The communication model is that one (1) Client communicates to a group of Servers with a single UPDATE request, as shown in Figure 34. Each Server receives the UPDATE request at approximately the same time and can execute the UPDATE request at approximately the same time. As example of this kind of communication is sending an “on” command to a group of lights, all lights that are member of that group turn on at approximately the same time. Sending UPDATE requests to a group of devices can be achieved on IP by means of sending messages to a predefined URL on a multicast address.

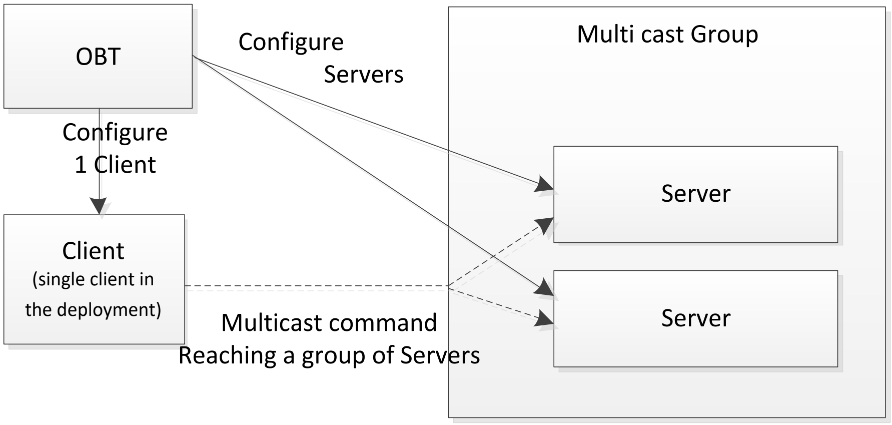

OBT
Configure Servers
Multi cast Group
Configure 1 Client
Server
Client
(single client in the deployment)
Server
Multicast command Reaching a group of Servers

Figure 34 – Simple Multicast requests

Security of SSM is accomplished by applying an application layer of in-transit protection and below the resource-access authorization layer, using OSCORE IETF RFC 8613.

Relative to an exchange of an UPDATE non-confirmable message:

- The Device acting as a Client (that is, sending an UPDATE request message) acts as an OSCORE client. Within the scope of clause 16.3 the single Client is assumed to support OSCORE and perform OSCORE client processing.
- The Device acting as a Server (that is, receiving an UPDATE request message) acts as an OSCORE server. Within the scope of clause 16.3, all Servers are assumed to support OSCORE and perform OSCORE server processing.

Clause 16.3.2 details the assumptions and prerequisites for correct functioning of SSM. Clause 16.3.3 describes the process for encapsulating an UPDATE request message into an SSM Request at the Client of an SSM Group, and subsequent extraction of an UPDATE request message from an SSM Request at the Server of an SSM Group. Clause 16.3.4 specifies how a Client generates an OSCORE Common Context and OSCORE Sender Context from an SSM Client Context and specifies how a Server generates an OSCORE Common Context and OSCORE Recipient Context from an SSM Server Context.

16.3.2 Assumptions and prerequisites for Simple Secure Multicast

As shown in the following example, any Server of the SSM Group can generate an SSM Request which other Servers in the SSM Group will interpret as being securely sent by the Client of the SSM Group, for the purposes of privilege escalation. The security of SSM relies on the assumption that no Server in the SSM Group attempts to generate an SSM Request using the credentials for the SSM Group. SSM should only be used in scenarios where the Security Domain Owner is confident that this is a valid assumption.

SSM Requests are delivered to SSM-capable Servers via the All OCF Nodes multicast address defined in ISO/IEC 30118-1. As specified in ISO/IEC 30118-1, all Servers subscribe to this multicast address to facilitate discovery of "oic/res", and consequently all Servers can receive SSM Requests delivered in this manner. A Server that supports the reception of SSM Requests for one or more Resources that it hosts shall populate the All OCF nodes multicast address in the "eps" Parameter of the Resource Links of those Resources in the "oic/res" discovery response.

The configured Client is aware of Multicast enabled Servers by means of detecting the multicast enabled resources in the Device discovery "oic/res" responses. The Client also knows how to create the multicast request to that resource, by means of the Introspection Device Data hosted on the Device. Therefore, the Client is able to send an UPDATE request to the multicast enabled Resources.

The Client of an SSM Group cannot form SSM Requests for the SSM Group until the Client is provisioned with the SSM Client Context for the SSM Group. Likewise, each Server in an SSM Group cannot process SSM Requests for the SSM Group until the Server is provisioned with the SSM Server Context for the SSM Group. The SSM Client Context and SSM Server Context are provisioned by an OBT as specified in OCF Onboarding Tool Specification. Clause 16.3.4 specifies how the OSCORE Sender Context at a Client is derived from an SSM Client Context, and how the OSCORE Recipient Context at a Server is derived from an SSM Server Context.

The UPDATE request encapsulated in an SSM Request includes a local URI path for a target Resource. A Server in the SSM Group for whom the request is intended, will process the request using the Resource at this local URI path, if such a Resource exists and the Resource matches the Resource Type and OCF Interface in the request. The SSM feature is designed with the assumption that the local URI path, Resource Type and supported OCF Interfaces on the intended Servers are consistent; but the SSM feature does not specify how such consistency is achieved.

The UPDATE request message itself is expected to contain information in such way that the Server can determine if the received UPDATE request message is intended for the Server, but the specification of this information is not part of the SSM feature.

16.3.3 OSCORE protection and verification of Simple Secure Multicast Requests

All OSCORE message processing requirements in clauses 8.1 and 8.2 in IETF RFC 8613 apply.

If a Client has an established SSM Client Context associated with an SSM Group, then the following call flow applies whenever the Client sends a multicast non-confirmable UPDATE request targeting multicast enabled Resources hosted on one or more Servers of the SSM Group.

1) The Client shall apply the OSCORE request protection processing to the UPDATE request as specified in clause 8.1 in IETF RFC 8613, using the OSCORE Security Context derived from the SSM Client Context as specified in clause 16.3.4. See ISO/IEC 30118-1 for details on setting the Proxy-URI option.

   The Client shall send the resulting OSCORE request message to the predefined All OCF Nodes multicast address. Dependent on the deployment scenario the different scopes as defined in clause 12.2.9 of ISO/IEC 30118-1 can be used.

2) All Servers subscribed to the predefined multicast address receive a copy of the OSCORE request message. Each Server supporting SSM which receives the OSCORE request message shall apply the OSCORE request verification and decryption processing in clause 8.2 of IETF RFC 8613 with the following clarifications:

   a) At Step 2 in clause 8.2 of IETF RFC 8613

      i) If either the decompression or the COSE message fails to decode, the Server shall ignore the message and shall not respond.

      ii) The Server attempts to retrieve the SSM Server Contexts with "recipientID" matching the ’kid’ parameter. If the Server fails to retrieve an SSM Server Context with "recipientID" matching the 'kid' parameter received, then the Server shall ignore the message and shall not respond.

   b) At step 6 in clause 8.2 of IETF RFC 8613, if the decryption failed then the Server shall ignore the message and shall not respond.

3) If any of the following criteria are met, then the CRUDN request message shall be silently discarded, and a response shall not be sent:

   – The operation of the CRUDN request is not the non-confirmable UDPATE operation on a multicast address.

   – The UPDATE request message is not intended for the Server – see clause 16.3.2 for further details.

   – There is no Resource hosted on the Server at the local URI path in the UPDATE request message.

4) The Server shall process the UPDATE request message (encapsulated in the OSCORE request message). The Server shall treat the value of "subjectuuid" in the credential entry which contains the OSCORE Security Context used to verify and decrypt the OSCORE request message in Step 2 as Client's Device UUID for access control processing. The Server shall treat the connection type as "auth-crypt" for access control processing. The Server shall not send a response.

The mechanism outlined is for sending a message in a send and forget mode, i.e. sending a message to a group of Servers, where each Server does not acknowledge the receipt. Since multicast requests are typically unreliable (e.g. non-confirmable messages) the best practice is to send the same UPDATE request more than once in a short time frame. This is sufficient since the multicast communication has in most cases a unicast variant for the same UPDATE request.

Notification (see clause 11.3 of ISO/IEC 30118-1) may be used to verify if the actual UPDATE request has been executed. If a subset of the group of Servers did not receive the UPDATE request, unicast (confirmable) messages can be used to complete the desired overall state of the system.

16.3.4 Creating OSCORE Security Context for Simple Secure Multicast

The present clause specifies how

- a Client of an SSM Group creates a OSCORE Security Context from a SSM Client Context provisioned to a credential entry of the Client.
- a Server of an SSM Group creates a OSCORE Security Context from a SSM Server Context provisioned to a credential entry of the Server.

All configurable parameters of the OSCORE Security Context are either:

- fixed to the OSCORE-specified default value, or
- directly provisioned by an OBT to a credential entry of the "/oic/sec/cred" Resource.

The following parameters of the OSCORE Security Context used for encryption by the Client of an SSM Group shall be set to the default values defined in clause 3.2 of IETF RFC 8613 (this information is not configured by the OBT):

- AEAD Algorithm,
- HKDF,
- Master Salt,
- ID Context.

The following parameters of the OSCORE Security Context parameters used for encryption by the Client of an SSM Group are derived from the SSM Client Context provisioned to a credential entry of "/oic/sec/cred" of the Client:

- The "subjectuuid" may be any schema compliant value. This Property serves no purpose when used in an SSM Client Context.
- The credential entry is identified as an SSM Client Context when the "credtype" matches the value specified for a SSM Client Context in Table 22, clause 13.3.1.
- The "privatedata" Property contains a 256-bit value which shall be used as the OSCORE Master Secret.
- The OSCORE Configuration parameters ("oscore") Property is present, and includes the following Properties:
  - The "senderid" Property shall be used as the OSCORE Sender ID of the OSCORE Security Context. The "recipientid" Property value shall be interpreted as the hexadecimal representation of a 56-bit value. The first byte of this value is expected to have the value assigned in Table Y for Simple Secure Multicast.
  - The "desc" Property is not used in security processing. This Property is described in clause 9.3.9.

On the Device shutting down, for each such credential entry, the Device shall write the value of corresponding OSCORE Sender Sequence Number as "ssn" Property to non-volatile memory. In event of a crash, devices should apply Appendix B.1.1 of IETF RFC 8613.

The following parameters of the OSCORE Security Context used by a Server of an SSM Group for verification and decryption shall be set to the default values defined in clause 3.2 of IETF RFC 8613 (this information is not configured by the OBT):

- AEAD Algorithm,
- HKDF,

- Replay Window,
- Master Salt,
- ID Context.

The following parameters of the OSCORE Security Context parameters used by a Server of an SSM Group for verification and decryption are derived from the SSM Server Context provisioned to a credential entry of "/oic/sec/cred" of the Server:

- The "subjectuuid" is used for access control processing as described in Step 4 of clause 16.3.3.
- The credential entry is identified as an SSM Server Context when the "credtype" matches to the value specified for an SSM Server Context in Table 22, clause 13.3.1.
- The "privatedata" Property of the credential entry contains a 256-bit value which shall be used as the OSCORE Master Secret.
- The OSCORE Configuration parameters ("oscore") Property is present, and includes the following Properties:
- The "recipientid" Property shall be used as the OSCORE Recipient ID of the OSCORE Security Context. The "recipientid" Property value shall be interpreted as the hexadecimal representation of a 56-bit value. The first byte of this value is expected to have the value assigned in Table Y for Simple Secure Multicast.
- The "desc" Property is not used in security processing. This Property is described in clause 9.3.9.

Annex A
(Informative)
Access Control Examples

Figure A-1 shows how an "/oic/sec/acl2" Resource could be configured to enforce an example access policy on the Server.

```
{
    "aclist2": [
      {
        // Subject with ID ...01 should access two named Resources with access mode "CRUDN" (Create, Retrieve, Update, Delete and Notify)
        "subject": {"uuid": "XXXX-...-XX01"},
        "resources": [
                      {"href":"/oic/sh/light/1"},
                      {"href":"/oic/sh/temp/0"}
  ],
        "permission": 31, // 31 dec = 0b0001 1111 which maps to ---N DURC
        "validity": [
           // The period starting at 18:00:00 UTC, on January 1, 2015 and
           // ending at 07:00:00 UTC on January 2, 2015
           "period": ["20150101T180000Z/20150102T070000Z"],
           // Repeats the {period} every week until the last day of Jan. 2015.
           "recurrence": ["RRULE:FREQ=WEEKLY;UNTIL=20150131T070000Z"]
          },
        "aceid": 1
      }
    ],
     // An ACL provisioning and management service should be identified as
     // the resource owner
    "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
}
```

Figure A-1 – Example "/oic/sec/acl2" Resource

Annex B
(Informative)
Execution environment security profiles

Given that IoT verticals and Devices will not be of uniform capabilities, a one-size-fits all security robustness requirements meeting all IOT applications and services will not serve the needs of OCF, and security profiles of varying degree of robustness (trustworthiness), cost and complexity have to be defined. To address a large ecosystem of vendors, the profiles can only be defined as requirements and the exact solutions meeting those requirements are specific to the vendors' open or proprietary implementations, and thus in most part outside scope of this document.

To align with the rest of OCF documents, where Device classifications follow IETF RFC 7228 (Terminology for constrained node networks) methodology, we limit the number of security profiles to a maximum of 3 (see Table B.1). However, our understanding is OCF capabilities criteria for each of 3 classes will be more fit to the current IoT chip market than that of IETF.

Given the extremely low level of resources at class 0, our expectation is that class 0 Devices are either capable of no security functionality or easily breakable security that depend on environmental (e.g. availability of human) factors to perform security functions. This means the class 0 will not be equipped with an SEE.

Table B.1 – OCF Security Profile

Platform class	SEE	Robustness level
0	No	N/A
1	Yes	Low
2	Yes	High

NOTE This analysis acknowledges that these Platform classifications do not take into consideration of possibility of security co-processor or other hardware security capability that augments classification criteria (namely CPU speed, memory, storage).

Annex C
(normative)
Resource Type definitions

C.1 List of Resource Type definitions

Table C.1 contains the list of defined security Resources in this document.

Table C.1 – Alphabetized list of security Resources

Friendly Name (informative)	Resource Type (rt)	Clause
Access Control List 2	oic.r.acl2	C.2
Auditable Event List	oic.r.ael	C.9
Certificate Signing Request	oic.r.csr	C.4
Credential	oic.r.cred	C.3
Device owner transfer method	oic.r.doxm	C.5
Device Provisioning Status	oic.r.pstat	C.6
Roles	oic.r.roles	C.7
Security Profile	oic.r.sp	C.8
Security Domain Information	oic.r.sdi	C.10

C.2 Access Control List-2

C.2.1 Introduction

This Resource specifies the local access control list.
When used without query parameters, all the ACE entries are returned.
When used with a query parameter, only the ACEs matching the specified parameter are returned.

C.2.2 Well-known URI

/oic/sec/acl2

C.2.3 Resource type

The Resource Type is defined as: "oic.r.acl2".

C.2.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Access Control List-2",
    "version": "2019-01-11",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "copyright 2016-2017, 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": {
    "/oic/sec/acl2" : {
      "get": {
        "description": "This Resource specifies the local access control list.\nWhen used without
query parameters, all the ACE entries are returned.\nWhen used with a query parameter, only the ACEs
matching the specified\nparameter are returned.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {"$ref": "#/parameters/ace-filtered"}
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example":
                {
                  "rt" : ["oic.r.acl2"],
                  "aclist2": [
                    {
                      "aceid": 1,
                      "subject": {
                        "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
                        "role": "SOME_STRING"
                      },
                      "resources": [
                        {
                          "href": "/light"
                        },
                        {
                          "href": "/door"
                        }
                      ],
                      "permission": 24
                    },
                    {
                      "aceid": 2,
                      "subject": {
                        "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
                      },
                      "resources": [
                        {
                          "href": "/light"
                        },
                        {
                          "href": "/door"
                        }
                      ],
                      "permission": 24
                      },
                      {
                        "aceid": 3,
                        "subject": {"conntype": "anon-clear"},
                        "resources": [
                          {
                            "href": "/light"
                          },
                          {
                            "href": "/door"
                          }
                        ],
                        "permission": 16,
                        "validity": [
                          {
                            "period":"20160101T180000Z/20170102T070000Z",
                            "recurrence": [ "DSTART:XXXXX",
"RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
                          },
                          {
                            "period":"20160101T180000Z/PT5H30M",
                            "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
                          }
                        ]
                      }
                  ],
                  "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
                },
              "schema": { "$ref": "#/definitions/Acl2" }
            },
            "400": {
              "description" : "The request is invalid."
            }
        }
      },
      "post": {
        "description": "Updates the ACL Resource with the provided ACEs.\n\nACEs provided in the
update with aceids not currently in the ACL\nResource are added.\n\nACEs provided in the update with
aceid(s) already in the ACL completely\nreplace the ACE(s) in the ACL Resource.\n\nACEs provided in
the update without aceid properties are added and\nassigned unique aceids in the ACL Resource.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {"$ref": "#/parameters/ace-filtered"},
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": { "$ref": "#/definitions/Acl2-Update" },
            "x-example":
              {
                "aclist2": [
                  {
                    "aceid": 1,
                    "subject": {
                      "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
                      "role": "SOME_STRING"
                    },
                    "resources": [
                      {
                        "href": "/light"
                      },
                      {
                        "href": "/door"
                      }
                    ],
                    "permission": 24
                  },
                  {
                    "aceid": 3,
                    "subject": {
                      "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
                    },
                    "resources": [
                      {
                        "href": "/light"
                      },
                      {
                        "href": "/door"
                      }
                    ],
                    "permission": 24
                  }
                ],
                "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
              }
          }
        ],
        "responses": {
            "400": {
              "description" : "The request is invalid."
            },
            "201": {
              "description" : "The ACL entry is created."
            },
            "204": {
              "description" : "The ACL entry is updated."
            }
        }
      },
      "delete": {
        "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
ACE entries are deleted.\nWhen DELETE is used with a query parameter, only the ACEs matching
the\nspecified parameter are deleted.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {"$ref": "#/parameters/ace-filtered"}
        ],
        "responses": {
            "200": {
              "description" : "The matching ACEs or the entire ACL Resource has been successfully
deleted."
            },
            "400": {
              "description" : "The request is invalid."
            }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : [ "oic.if.rw", "oic.if.baseline" ]
    },
    "ace-filtered" : {
      "in" : "query",
      "name" : "aceid",
      "required" : false,
      "type" : "integer",
      "description" : "Only applies to the ACE with the specified aceid.",
      "x-example" : 2112
    }
  },
  "definitions": {
    "Acl2" : {
      "properties": {
        "rowneruuid" : {
          "description": "The value identifies the unique Resource owner\nFormat pattern according
to IETF RFC 4122.",
          "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}$",
          "type": "string"
        },
        "rt" : {
          "description": "Resource Type of the Resource.",
          "items": {
            "maxLength": 64,
            "type": "string",
            "enum": ["oic.r.acl2"]
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "aclist2" : {
          "description": "Access Control Entries in the ACL Resource.",
          "items": {
            "properties": {
              "aceid": {
                "description": "An identifier for the ACE that is unique within the ACL. In cases
where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
                "minimum": 1,
                "type": "integer"
              },
              "permission": {
                "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
permissions.",
                "x-detail-desc": [
                  "0  - No permissions",
                  "1 - Create permission is granted",
                  "2 - Read, observe, discover permission is granted",
                  "4 - Write, update permission is granted",
                  "8 - Delete permission is granted",
                  "16 - Notify permission is granted"
                ],
                "maximum": 31,
                "minimum": 0,
                "type": "integer"
              },
              "resources": {
                "description": "References the application's Resources to which a security policy
applies.",
                "items": {
                  "description": "Each Resource must have at least one of these properties set.",
                  "properties": {
                    "href": {
                      "description": "When present, the ACE only applies when the href matches\nThis
is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
                      "format": "uri",
                      "maxLength": 256,
                      "type": "string"
                    },
                    "wc": {
                      "description": "A wildcard matching policy.",
                      "pattern": "^[-+*]$",
                      "type": "string"
                    }
                  },
                  "type": "object"
                },
                "type": "array"
              },
              "subject": {
                "anyOf": [
                  {
                    "description": "This is the Device identifier.",
                    "properties": {
                      "uuid": {
                        "description": "A UUID Device ID\nFormat pattern according to IETF RFC
4122.",
                        "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}$",
                        "type": "string"
                      }
                    },
                    "required": [
                      "uuid"
                    ],
                    "type": "object"
                  },
                  {
                    "description": "Security role specified as an <Authority> & <Rolename>. A NULL
<Authority> refers to the local entity or Device.",
                    "properties": {
                      "authority": {
                        "description": "The Authority component of the entity being identified. A
NULL <Authority> refers to the local entity or Device.",
                        "type": "string"
                      },
                      "role": {
                        "description": "The ID of the role being identified.",
                        "type": "string"
                      }
                    },
                    "required": [
                      "role"
                    ],
                    "type": "object"
                  },
                  {
                    "properties": {
                      "conntype": {
                        "description": "This property allows an ACE to be matched based on the
connection or message type.",
                        "x-detail-desc": [
                          "auth-crypt - ACE applies if the Client is authenticated and the data
channel or message is encrypted and integrity protected",
                          "anon-clear - ACE applies if the Client is not authenticated and the data
channel or message is not encrypted but may be integrity protected"
                        ],
                        "enum": [
                          "auth-crypt",
                          "anon-clear"
                        ],
                        "type": "string"
                      }
                    },
                    "required": [
                      "conntype"
                    ],
                    "type": "object"
                  }
                ]
              },
              "validity": {
                "description": "validity is an array of time-pattern objects.",
                "items": {
                  "description": "The time-pattern contains a period and recurrence expressed in
RFC5545 syntax.",
                  "properties": {
                    "period": {
                      "description": "String represents a period using the RFC5545 Period.",
                      "type": "string"
                    },
                    "recurrence": {
                      "description": "String array represents a recurrence rule using the RFC5545
Recurrence.",
                      "items": {
                        "type": "string"
                      },
                      "type": "array"
                    }
                  },
                  "required": [
                    "period"
                  ],
                  "type": "object"
                },
                "type": "array"
              }
            },
            "required": [
              "aceid",
              "resources",
              "permission",
              "subject"
            ],
            "type": "object"
          },
          "type": "array"
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "if" : {
          "description": "The interface set supported by this Resource.",
          "items": {
            "enum": [ "oic.if.rw", "oic.if.baseline" ],
            "type": "string"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        }
      },
      "type" : "object",
      "required": ["aclist2", "rowneruuid"]
    },
    "Acl2-Update" : {
      "properties": {
        "rowneruuid" : {
           "description": "The value identifies the unique Resource owner\n Format pattern according to IETF RFC 4122.",
           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
           "type": "string"
        },
        "aclist2" : {
          "description": "Access Control Entries in the ACL Resource.",
          "items": {
            "properties": {
              "aceid": {
                "description": "An identifier for the ACE that is unique within the ACL. In cases where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
                "minimum": 1,
                "type": "integer"
              },
              "permission": {
                "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating permissions.",
                "x-detail-desc": [
                  "0  - No permissions",
                  "1 - Create permission is granted",
                  "2 - Read, observe, discover permission is granted",
                  "4 - Write, update permission is granted",
                  "8 - Delete permission is granted",
                  "16 - Notify permission is granted"
                ],
                "maximum": 31,
                "minimum": 0,
                "type": "integer"
              },
              "resources": {
                "description": "References the application's Resources to which a security policy applies.",
                "items": {
                  "description": "Each Resource must have at least one of these properties set.",
                  "properties": {
                    "href": {
                      "description": "When present, the ACE only applies when the href matches\nThis is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
                      "format": "uri",
                      "maxLength": 256,
                      "type": "string"
                    },
                    "wc": {
                      "description": "A wildcard matching policy.",
                      "x-detail-desc": [
                        "+ - Matches all discoverable Resources",
                        "- - Matches all non-discoverable Resources",
                        "* - Matches all Resources"
                      ],
                      "enum": [
                        "+",
                        "-",
                        "*"
                      ],
                      "type": "string"
                    }
                  },
                  "type": "object"
                },
                "type": "array"
              },
              "subject": {
                "anyOf": [
                  {
                    "description": "This is the Device identifier.",
                    "properties": {
                      "uuid": {
                        "description": "A UUID Device ID\n Format pattern according to IETF RFC 4122.",
                        "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}$",
                        "type": "string"
                      }
                    },
                    "required": [
                      "uuid"
                    ],
                    "type": "object"
                  },
                  {
                    "description": "Security role specified as an <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
                    "properties": {
                      "authority": {
                        "description": "The Authority component of the entity being identified. A NULL <Authority> refers to the local entity or Device.",
                        "type": "string"
                      },
                      "role": {
                        "description": "The ID of the role being identified.",
                        "type": "string"
                      }
                    },
                    "required": [
                      "role"
                    ],
                    "type": "object"
                  },
                  {
                    "properties": {
                      "conntype": {
                        "description": "This property allows an ACE to be matched based on the connection or message type.",
                        "x-detail-desc": [
                          "auth-crypt - ACE applies if the Client is authenticated and the data channel or message is encrypted and integrity protected",
                          "anon-clear - ACE applies if the Client is not authenticated and the data channel or message is not encrypted but may be integrity protected"
                        ],
                        "enum": [
                          "auth-crypt",
                          "anon-clear"
                        ],
                        "type": "string"
                      }
                    },
                    "required": [
                      "conntype"
                    ],
                    "type": "object"
                  }
                ]
              },
              "validity": {
                "description": "validity is an array of time-pattern objects.",
                "items": {
                  "description": "The time-pattern contains a period and recurrence expressed in
RFC5545 syntax.",
                  "properties": {
                    "period": {
                      "description": "String represents a period using the RFC5545 Period.",
                      "type": "string"
                    },
                    "recurrence": {
                      "description": "String array represents a recurrence rule using the RFC5545
Recurrence.",
                      "items": {
                        "type": "string"
                      },
                      "type": "array"
                    }
                  },
                  "required": [
                    "period"
                  ],
                  "type": "object"
                },
                "type": "array"
              }
            },
            "required": [
              "resources",
              "permission",
              "subject"
            ],
            "type": "object"
          },
          "type": "array"
        }
      },
      "type" : "object"
    }
  }
}
```

C.2.5 Property definition

Table C-1 defines the Properties that are part of the "oic.r.acl2" Resource Type.

Table C-1 – The Property definitions of the Resource with type "rt" = "oic.r.acl2".

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	Yes	Read Write	The value identifies the unique Resource owner Format pattern according to IETF RFC 4122.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
aclist2	array: see schema	Yes	Read Write	Access Control Entries in the ACL Resource.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The interface set supported by this Resource.

rowneruuid	string	No	Read Write	The value identifies the unique Resource owner Format pattern according to IETF RFC 4122.
aclist2	array: see schema	No	Read Write	Access Control Entries in the ACL Resource.

C.2.6 CRUDN behaviour

Table C-2 defines the CRUDN operations that are supported on the "oic.r.acl2" Resource Type.

Table C-2 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl2".

Create	Read	Update	Delete	Notify
	get	post	delete	observe

C.3 Credential

C.3.1 Introduction

This Resource specifies credentials a Device may use to establish secure communication.
Retrieves the credential data.
When used without query parameters, all the credential entries are returned.
When used with a query parameter, only the credentials matching the specified parameter are returned.

Note that write-only credential data will not be returned.

C.3.2 Well-known URI

/oic/sec/cred

C.3.3 Resource type

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

C.3.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Credential",
    "version": "2020-10-19",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "copyright 2016-2017, 2019, 2020 Open Connectivity Foundation, Inc. All rights
reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": ["http"],
  "consumes": ["application/json"],
  "produces": ["application/json"],
  "paths": {
    "/oic/sec/cred": {
      "get": {
        "description": "This Resource specifies credentials a Device may use to establish secure
communication.\nRetrieves the credential data.\nWhen used without query parameters, all the
credential entries are returned.\nWhen used with a query parameter, only the credentials matching
```

```
the specified\nparameter are returned.\n\nNote that write-only credential data will not be
returned.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {"$ref": "#/parameters/cred-filtered-credid"},
          {"$ref": "#/parameters/cred-filtered-subjectuuid"}
        ],
        "responses": {
            "200": {
              "description": "",
              "x-example": {
                  "rt": ["oic.r.cred"],
                  "creds": [
                    {
                      "credid": 55,
                      "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
                      "roleid": {
                        "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
                        "role": "SOME_STRING"
                      },
                      "credtype": 32,
                      "publicdata": {
                        "encoding": "oic.sec.encoding.pem",
                        "data": "PEM-ENCODED-VALUE"
                      },
                      "privatedata": {
                        "encoding": "oic.sec.encoding.raw",
                        "data": "RAW-ENCODED-VALUE",
                        "handle": 4
                      },
                      "optionaldata": {
                        "revstat": false,
                        "encoding": "oic.sec.encoding.pem",
                        "data": "PEM-ENCODED-VALUE"
                      },
                      "period": "20160101T180000Z/20170102T070000Z",
                      "crms": [ "oic.sec.crm.pk10" ]
                    },
                    {
                      "credid": 56,
                      "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
                      "roleid": {
                        "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
                        "role": "SOME_STRING"
                      },
                      "credtype": 1,
                      "publicdata": {
                        "encoding": "oic.sec.encoding.pem",
                        "data": "PEM-ENCODED-VALUE"
                      },
                      "privatedata": {
                        "encoding": "oic.sec.encoding.base64",
                        "data": "BASE-64-ENCODED-VALUE",
                        "handle": 4
                      },
                      "optionaldata": {
                        "revstat": false,
                        "encoding": "oic.sec.encoding.pem",
                        "data": "PEM-ENCODED-VALUE"
                      },
                      "period": "20160101T180000Z/20170102T070000Z",
                      "crms": [ "oic.sec.crm.pk10" ]
                    }
                  ],
                  "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
                }
                ,
              "schema": { "$ref": "#/definitions/Cred" }
            },
            "400": {
              "description": "The request is invalid."
            }
        }
      },
      "post": {
        "description": "Updates the credential Resource with the provided
credentials.\n\nCredentials provided in the update with credid(s) not currently in the\ncredential
Resource are added.\n\nCredentials provided in the update with credid(s) already in the\ncredential
Resource completely replace the creds in the credential\nResource.\n\nCredentials provided in the
update without credid(s) properties are\nadded and assigned unique credid(s) in the credential
Resource.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": { "$ref": "#/definitions/Cred-Update" },
            "x-example":
              {
                "creds": [
                  {
                    "credid": 55,
                    "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
                    "roleid": {
                      "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
                      "role": "SOME_STRING"
                    },
                    "credtype": 32,
                    "publicdata": {
                      "encoding": "oic.sec.encoding.pem",
                      "data": "PEM-ENCODED-VALUE"
                    },
                    "privatedata": {
                      "encoding": "oic.sec.encoding.raw",
                      "data": "RAW-ENCODED-VALUE",
                      "handle": 4
                    },
                    "optionaldata": {
                      "revstat": false,
                      "encoding": "oic.sec.encoding.pem",
                      "data": "PEM-ENCODED-VALUE"
                    },
                    "period": "20160101T180000Z/20170102T070000Z",
                    "crms": [ "oic.sec.crm.pk10" ]
                  },
                  {
                    "credid": 56,
                    "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
                    "roleid": {
                      "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
                      "role": "SOME_STRING"
                    },
                    "credtype": 1,
                    "publicdata": {
                      "encoding": "oic.sec.encoding.pem",
                      "data": "PEM-ENCODED-VALUE"
                    },
                    "privatedata": {
                      "encoding": "oic.sec.encoding.base64",
                      "data": "BASE-64-ENCODED-VALUE",
                      "handle": 4
                    },
                    "optionaldata": {
                      "revstat": false,
                      "encoding": "oic.sec.encoding.pem",
                      "data": "PEM-ENCODED-VALUE"
                    },
                    "period": "20160101T180000Z/20170102T070000Z",
                    "crms": [ "oic.sec.crm.pk10" ]
                  }
                ],
                "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
              }
          }
        ],
        "responses": {
            "400": {
              "description": "The request is invalid."
            },
            "201": {
              "description": "The credential entry is created."
            },
            "204": {
              "description": "The credential entry is updated."
            }
        }
      },
      "delete": {
        "description": "Deletes credential entries.\nWhen DELETE is used without query parameters,
all the cred entries are deleted.\nWhen DELETE is used with a query parameter, only the entries
matching\nthe query parameter are deleted.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {"$ref": "#/parameters/cred-filtered-credid"},
          {"$ref": "#/parameters/cred-filtered-subjectuuid"}
        ],
        "responses": {
            "400": {
              "description": "The request is invalid."
            },
            "204": {
              "description": "The specific credential(s) or the the entire credential Resource has
been successfully deleted."
            }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : [ "oic.if.rw", "oic.if.baseline" ]
    },
    "cred-filtered-credid": {
      "in": "query",
      "name": "credid",
      "required": false,
      "type": "integer",
      "description": "Only applies to the credential with the specified credid.",
      "x-example": 2112
    },
    "cred-filtered-subjectuuid": {
      "in": "query",
      "name": "subjectuuid",
      "required": false,
      "type": "string",
      "description": "Only applies to credentials with the specified subject UUID.",
      "x-example": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
    }
  },
  "definitions": {
    "Cred": {
      "properties": {
        "rowneruuid": {
          "description": "Format pattern according to IETF RFC 4122.",
          "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9]{12}$",
          "type": "string"
        },
        "rt": {
          "description": "Resource Type of the Resource.",
          "items": {
            "maxLength": 64,
            "type": "string",
            "enum": ["oic.r.cred"]
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array",
          "uniqueItems": true
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "creds": {
          "description": "List of credentials available at this Resource.",
          "items": {
            "properties": {
              "credid": {
                "description": "Local reference to a credential Resource.",
                "type": "integer"
              },
              "credtype": {
                 "description": "Representation of this credential's type\nCredential Types - Cred
type encoded as a bitmask.0 - Empty credential used for testing\n1 - Symmetric pair-wise key\n2 -
Symmetric group key\n4 - Asymmetric signing key\n8 - Asymmetric signing key with certificate\n16 -
PIN or password\n32 - Asymmetric encryption key. \n 128 - SSM Client\n256 - SSM Server",
                 "maximum": 256,
                 "minimum": 0,
                 "type": "integer"
              },
              "credusage": {
                "description": "A string that provides hints about how/where the cred is used\nThe
type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
                "enum": [
                  "oic.sec.cred.trustca",
                  "oic.sec.cred.cert",
                  "oic.sec.cred.rolecert",
                  "oic.sec.cred.mfgtrustca",
                  "oic.sec.cred.mfgcert"
                ],
                "type": "string"
              },
              "crms": {
                "description": "The refresh methods that may be used to update this credential.",
                "items": {
                  "description": "Each enum represents a method by which the credentials are
refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
                  "enum": [
                    "oic.sec.crm.pro",
                    "oic.sec.crm.psk",
                    "oic.sec.crm.rdp",
                    "oic.sec.crm.skdc",
                    "oic.sec.crm.pk10"
                  ],
                  "type": "string"
                },
                "type": "array",
                "uniqueItems": true
              },
              "optionaldata": {
                "description": "Credential Type dependent. Credential revocation status
information\n1, 2, 4, 32, 64: revocation status information\n8: Revocation information",
                "properties": {
                  "data": {
                    "description": "The encoded structure.",
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the optdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain."
                    ],
                    "enum": [
                      "oic.sec.encoding.pem"
                    ],
                    "type": "string"
                  },
                  "revstat": {
                    "description": "Revocation status flag - true = revoked.",
                    "type": "boolean"
                  }
                },
                "required": [
                  "revstat"
                ],
                "type": "object"
              },
              "period": {
                "description": "String with RFC5545 Period.",
                "type": "string"
              },
              "privatedata": {
                "description": "Private credential information\nCredential Resource non-public
contents.",
                "properties": {
                  "data": {
                    "description": "The encoded value.",
                    "maxLength": 3072,
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the privdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.pem - Encoding for PEM encoded private key.",
                      "oic.sec.encoding.base64 - Encoding for Base64 encoded PSK.",
                      "oic.sec.encoding.handle - Data is contained in a storage sub-system
referenced using a handle.",
                      "oic.sec.encoding.raw - Raw hex encoded data."
                    ],
                    "enum": [
                      "oic.sec.encoding.pem",
                      "oic.sec.encoding.base64",
                      "oic.sec.encoding.handle",
                      "oic.sec.encoding.raw"
                    ],
                    "type": "string"
                  },
                  "handle": {
                    "description": "Handle to a key storage Resource.",
                    "type": "integer"
                  }
                },
                "required": [
                  "encoding"
                ],
                "type": "object"
              },
              "publicdata": {
                "description": "Credential Type dependent. Public credential information\n1:2:
ticket, public SKDC values\n4, 32: Public key value\n8: A chain of one or more certificate",
                "properties": {
                  "data": {
                    "description": "The encoded value.",
                    "maxLength": 3072,
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the pubdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain."
                    ],
                    "enum": [
                      "oic.sec.encoding.pem"
                    ],
                    "type": "string"
                  }
                },
                "type": "object"
              },
              "oscore": {
                "description": "Contains parameters for use with credentials intended for use with
OSCORE. See type definition for \"oic.sec.oscoretype\"",
                "properties": {
                  "senderid": {
                    "description": "OSCORE Sender ID for this OSCORE Security Context",
                    "type": "string"
                  },
                  "recipientid": {
                    "description": "OSCORE Recipient ID for this OSCORE Security Context",
                    "type": "string"
                  },
                  "ssn": {
                    "description": "OSCORE Sender Sequence Number SSN1 being stored in nonvolatile
memory to handle the loss of mutable security context parameters",
                    "type": "integer",
                    "readOnly": true
                  },
                  "desc": {
                    "description": "Human readable description of the usage of this OSCORE Security
Context",
                    "type": "string"
                  }
                },
                "type": "object"
              },
              "roleid": {
                "description": "The role this credential possesses\nSecurity role specified as an
<Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
                "properties": {
                  "authority": {
                    "description": "The Authority component of the entity being identified. A NULL
<Authority> refers to the local entity or Device.",
                    "type": "string"
                  },
                  "role": {
                    "description": "The ID of the role being identified.",
                    "type": "string"
                  }
                },
                "required": [
                  "role"
                ],
                "type": "object"
              },
              "subjectuuid": {
                "anyOf": [
                  {
                    "description": "The id of the Device, which the cred entry applies to or \"*\"
for wildcard identity.",
                    "pattern": "^\\*$",
                    "type": "string"
                  },
                  {
                    "description": "Format pattern according to IETF RFC 4122.",
                    "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
F0-9]{12}$",
                    "type": "string"
                  }
                ]
              }
            },
            "type": "object"
          },
          "type": "array"
        },
        "if": {
          "description": "The interface set supported by this Resource.",
          "items": {
            "enum": [ "oic.if.rw", "oic.if.baseline" ],
            "type": "string"
          },
          "minItems": 2,
          "readOnly": true,
          "type": "array"
        }
      },
      "type": "object",
      "required": ["creds", "rowneruuid"]
    },
    "Cred-Update": {
      "properties": {
        "rowneruuid": {
          "description": "Format pattern according to IETF RFC 4122.",
          "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}$",
          "type": "string"
        },
        "creds": {
          "description": "List of credentials available at this Resource.",
          "items": {
            "properties": {
              "credid": {
                "description": "Local reference to a credential Resource.",
                "type": "integer"
              },
              "credtype": {
                "description": "Representation of this credential's type\nCredential Types - Cred
type encoded as a bitmask.0 - Empty credential used for testing\n1 - Symmetric pair-wise key\n2 -
Symmetric group key\n4 - Asymmetric signing key\n8 - Asymmetric signing key with certificate\n16 -
PIN or password\n32 - Asymmetric encryption key. \n 128 - SSM Client\n256 - SSM Server",
                "maximum": 256,
                "minimum": 0,
                "type": "integer"
              },
              "credusage": {
                "description": "A string that provides hints about how/where the cred is used\nThe
type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
                "enum": [
                  "oic.sec.cred.trustca",
                  "oic.sec.cred.cert",
                  "oic.sec.cred.rolecert",
                  "oic.sec.cred.mfgtrustca",
                  "oic.sec.cred.mfgcert"
                ],
                "type": "string"
              },
              "crms": {
                "description": "The refresh methods that may be used to update this credential.",
                "items": {
                  "description": "Each enum represents a method by which the credentials are
refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
```

```
serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
                  "enum": [
                    "oic.sec.crm.pro",
                    "oic.sec.crm.psk",
                    "oic.sec.crm.rdp",
                    "oic.sec.crm.skdc",
                    "oic.sec.crm.pk10"
                  ],
                  "type": "string"
                },
                "type": "array"
              },
              "optionaldata": {
                "description": "Credential Type dependent. Credential revocation status
information\n1, 2, 4, 32, 64: revocation status information\n8: Revocation information",
                "properties": {
                  "data": {
                    "description": "The encoded structure.",
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the optdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain."
                    ],
                    "enum": [
                      "oic.sec.encoding.pem"
                    ],
                    "type": "string"
                  },
                  "revstat": {
                    "description": "Revocation status flag - true = revoked.",
                    "type": "boolean"
                  }
                },
                "required": [
                  "revstat"
                ],
                "type": "object"
              },
              "period": {
                "description": "String with RFC5545 Period.",
                "type": "string"
              },
              "privatedata": {
                "description": "Private credential information\nCredential Resource non-public
contents.",
                "properties": {
                  "data": {
                    "description": "The encoded value.",
                    "maxLength": 3072,
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the privdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.pem - Encoding for PEM encoded private key.",
                      "oic.sec.encoding.base64 - Encoding for Base64 encoded PSK.",
                      "oic.sec.encoding.handle - Data is contained in a storage sub-system
referenced using a handle.",
                      "oic.sec.encoding.raw - Raw hex encoded data."
                    ],
                    "enum": [
                      "oic.sec.encoding.pem",
                      "oic.sec.encoding.base64",
                      "oic.sec.encoding.handle",
                      "oic.sec.encoding.raw"
                    ],
                    "type": "string"
                  },
                  "handle": {
                    "description": "Handle to a key storage Resource.",
                    "type": "integer"
                  }
                },
                "required": [
                  "encoding"
                ],
                "type": "object"
              },
              "publicdata": {
                "description": "Credential Type dependent. Public credential information\n1:2:
ticket, public SKDC values\n4, 32: Public key value\n8: A chain of one or more certificate",
                "properties": {
                  "data": {
                    "description": "The encoded value.",
                    "maxLength": 3072,
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the pubdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain."
                    ],
                    "enum": [
                      "oic.sec.encoding.pem"
                    ],
                    "type": "string"
                  }
                },
                "type": "object"
              },
              "oscore": {
                "description": "Contains parameters for use with credentials intended for use with
OSCORE. See type definition for \"oic.sec.oscoretype\"",
                "properties": {
                  "senderid": {
                    "description": "OSCORE Sender ID for this OSCORE Security Context",
                    "type": "string"
                  },
                  "recipientid": {
                    "description": "OSCORE Recipient ID for this OSCORE Security Context",
                    "type": "string"
                  },
                  "desc": {
                    "description": "Human readable description of the usage of this OSCORE Security
Context",
                    "type": "string"
                  }
                },
                "type": "object"
              },
              "roleid": {
                "description": "The role this credential possesses\nSecurity role specified as an
<Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
                "properties": {
                  "authority": {
                    "description": "The Authority component of the entity being identified. A NULL
<Authority> refers to the local entity or Device.",
                    "type": "string"
                  },
                  "role": {
                    "description": "The ID of the role being identified.",
                    "type": "string"
                  }
                },
                "required": [
                  "role"
                ],
                "type": "object"
              },
              "subjectuuid": {
                "anyOf": [
                  {
                    "description": "The id of the Device, which the cred entry applies to or \"*\"
for wildcard identity.",
                    "pattern": "^\\*$",
                    "type": "string"
                  },
                  {
                    "description": "Format pattern according to IETF RFC 4122.",
                    "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
F0-9]{12}$",
                    "type": "string"
                  }
                ]
              }
            },
            "type": "object"
          },
          "type": "array"
        },
        "if": {
          "description": "The interface set supported by this Resource.",
          "items": {
            "enum": [ "oic.if.rw", "oic.if.baseline" ],
            "type": "string"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        }
      },
      "type": "object"
    }
  }
}
```

C.3.5 Property definition

Table C-3 defines the Properties that are part of the "oic.r.cred" Resource Type.

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

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
creds	array: see schema	Yes	Read Write	List of credentials available at this Resource.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
rowneruuid	string	No	Read Write	Format pattern according to IETF RFC 4122.

creds	array: see schema	No	Read Write	List of credentials available at this Resource.
if	array: see schema	No	Read Only	The interface set supported by this Resource.

C.3.6 CRUDN behaviour

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

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

Create	Read	Update	Delete	Notify
	get	post	delete	observe

C.4 Certificate Signing Request

C.4.1 Introduction

This Resource specifies a Certificate Signing Request.

C.4.2 Well-known URI

/oic/sec/csr

C.4.3 Resource type

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

C.4.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Certificate Signing Request",
    "version": "2015-08-19",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "copyright 2016-2017, 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": {
    "/oic/sec/csr" : {
      "get": {
        "description": "This Resource specifies a Certificate Signing Request.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"}
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example":
                {
                "rt": ["oic.r.csr"],
                "encoding" : "oic.sec.encoding.pem",
                "csr": "PEMENCODEDCSR"
                },
              "schema": { "$ref": "#/definitions/Csr" }
            },
            "404": {
              "description" : "The Device does not support certificates and generating CSRs."
            },
            "503": {
              "description" : "The Device is not yet ready to return a response. Try again later."
            }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : [ "oic.if.rw", "oic.if.baseline" ]
    }
  },
  "definitions": {
    "Csr" : {
      "properties": {
        "rt" : {
          "description": "Resource Type of the Resource.",
          "items": {
            "maxLength": 64,
            "type": "string",
            "enum": ["oic.r.csr"]
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "encoding": {
          "description": "A string specifying the encoding format of the data contained in CSR.",
          "x-detail-desc": [
            "oic.sec.encoding.pem - Encoding for PEM encoded CSR."
          ],
          "enum": [
            "oic.sec.encoding.pem"
          ],
          "readOnly": true,
          "type": "string"
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "csr": {
          "description": "Signed CSR in ASN.1 in the encoding specified by the encoding property.",
          "maxLength": 3072,
          "readOnly": true,
          "type": "string"
        },
        "if": {
          "description": "The interface set supported by this Resource.",
          "items": {
            "enum": [ "oic.if.rw", "oic.if.baseline" ],
            "type": "string"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        }
      },
      "type" : "object",
      "required": ["csr", "encoding"]
    }
  }
}
```

C.4.5 Property definition

Table C-5 defines the Properties that are part of the "oic.r.csr" Resource Type.

Table C-5 – The Property definitions of the Resource with type "rt" = "oic.r.csr".

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource.
encoding	string	Yes	Read Only	A string specifying the encoding format of the data contained in CSR.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
csr	string	Yes	Read Only	Signed CSR in ASN.1 in the encoding specified by the encoding property.
if	array: see schema	No	Read Only	The interface set supported by this Resource.

C.4.6 CRUDN behaviour

Table C-6 defines the CRUDN operations that are supported on the "oic.r.csr" Resource Type.

Table C-6 – The CRUDN operations of the Resource with type "rt" = "oic.r.csr".

Create	Read	Update	Delete	Notify
	get			observe

C.5 Device Owner Transfer Method

C.5.1 Introduction

This Resource specifies properties needed to establish a Device owner.

C.5.2 Well-known URI

/oic/sec/doxm

C.5.3 Resource type

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

C.5.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Device Owner Transfer Method",
    "version": "2020-10-19",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "copyright 2016-2017, 2019, 2020 Open Connectivity Foundation, Inc. All rights
reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": ["http"],
  "consumes": ["application/json"],
  "produces": ["application/json"],
  "paths": {
    "/oic/sec/doxm" : {
      "get": {
        "description": "This Resource specifies properties needed to establish a Device owner.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {"$ref": "#/parameters/owned"}
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example": {
                  "rt": ["oic.r.doxm"],
                  "oxms": [ 0, 2, 3 ],
                  "oxmsel": 0,
                  "sct": 16,
                  "owned": true,
                  "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
                  "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
                  "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
                },
              "schema": { "$ref": "#/definitions/Doxm" }
            },
            "400": {
              "description" : "The request is invalid."
            }
        }
      },
      "post": {
        "description": "Updates the DOXM Resource data.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": { "$ref": "#/definitions/Doxm-Update" },
            "x-example":
              {
                "oxmsel": 0,
                "owned": true,
                "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
                "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
                "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
              }
          }
        ],
        "responses": {
            "400": {
              "description" : "The request is invalid."
            },
            "204": {
              "description" : "The DOXM entry is updated."
            }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : [ "oic.if.rw", "oic.if.baseline" ]
    },
    "owned": {
      "in": "query",
      "name": "owned",
      "type": "boolean"
    }
  },
  "definitions": {
    "Doxm" : {
      "properties": {
        "rowneruuid": {
          "description": "Format pattern according to IETF RFC 4122.",
          "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9]{12}$",
          "type": "string"
        },
        "oxms": {
          "description": "List of supported owner transfer methods.",
          "items": {
            "description": "The Device owner transfer methods that may be selected at Device on-
boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the
Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method
(oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method
(oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap)
(deprecated).",
            "type": "integer"
          },
          "readOnly": true,
          "type": "array"
        },
        "devowneruuid": {
          "description": "Format pattern according to IETF RFC 4122.",
          "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}$",
          "type": "string"
        },
        "deviceuuid": {
          "description": "The uuid formatted identity of the Device\nFormat pattern according to
IETF RFC 4122.",
          "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9]{12}$",
          "type": "string"
        },
        "owned": {
          "description": "Ownership status flag.",
          "type": "boolean"
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "oxmsel": {
              "description": "The selected owner transfer method used during on-boarding\nThe Device
owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
method (oic.sec.doxm.dcap) (deprecated).",
              "type": "integer"
        },
        "sct": {
              "description": "Bitmask encoding of supported credential types\nCredential Types -
Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 - Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or password32 - Asymmetric encryption key.",
              "maximum": 511,
              "minimum": 0,
              "type": "integer",
              "readOnly": true
        },
        "rt": {
          "description": "Resource Type of the Resource.",
          "items": {
            "maxLength": 64,
            "type": "string",
            "enum": ["oic.r.doxm"]
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "if": {
          "description": "The OCF Interface set supported by this Resource.",
          "items": {
            "enum": [ "oic.if.rw", "oic.if.baseline" ],
            "type": "string"
          },
          "minItems": 2,
          "readOnly": true,
          "type": "array"
        }
      },
      "type" : "object",
      "required": ["oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid", "rowneruuid"]
    },
    "Doxm-Update" : {
      "properties": {
        "rowneruuid": {
          "description": "Format pattern according to IETF RFC 4122.",
          "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}$",
          "type": "string"
        },
        "devowneruuid": {
          "description": "Format pattern according to IETF RFC 4122.",
          "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
          "type": "string"
        },
        "deviceuuid": {
              "description": "The uuid formatted identity of the Device\nFormat pattern according to IETF RFC 4122.",
              "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}$",
              "type": "string"
        },
        "owned": {
          "description": "Ownership status flag.",
          "type": "boolean"
        },
        "oxmsel": {
              "description": "The selected owner transfer method used during on-boarding\nThe Device owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap) (deprecated).",
              "type": "integer"
        }
      },
      "type" : "object"
    }
  }
```

```
}
```

C.5.5 Property definition

Table C-7 defines the Properties that are part of the "oic.r.doxm" Resource Type.

Table C-7 – The Property definitions of the Resource with type "rt" = "oic.r.doxm".

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
oxms	array: see schema	Yes	Read Only	List of supported owner transfer methods.
devowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
deviceuuid	string	Yes	Read Write	The uuid formatted identity of the Device Format pattern according to IETF RFC 4122.
owned	boolean	Yes	Read Write	Ownership status flag.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
oxmsel	integer	Yes	Read Write	The selected owner transfer method used during on-boarding The Device owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap) (deprecated).
sct	integer	Yes	Read Only	Bitmask encoding of supported credential types Credential Types - Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 - Symmetric group key4 -

				Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or password32 - Asymmetric encryption key.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
if	array: see schema	No	Read Only	The OCF Interface set supported by this Resource.
rowneruuid	string		Read Write	Format pattern according to IETF RFC 4122.
devowneruuid	string		Read Write	Format pattern according to IETF RFC 4122.
deviceuuid	string		Read Write	The uuid formatted identity of the Device Format pattern according to IETF RFC 4122.
owned	boolean		Read Write	Ownership status flag.
oxmsel	integer		Read Write	The selected owner transfer method used during on-boarding The Device owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap) (deprecated).

C.5.6 CRUDN behaviour

Table C-8 defines the CRUDN operations that are supported on the "oic.r.doxm" Resource Type.

Table C-8 – The CRUDN operations of the Resource with type "rt" = "oic.r.doxm".

Create	Read	Update	Delete	Notify
	get	post		observe

C.6 Device Provisioning Status

C.6.1 Introduction

This Resource specifies Device provisioning status.

C.6.2 Well-known URI

/oic/sec/pstat

C.6.3 Resource type

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

C.6.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Device Provisioning Status",
    "version": "2019-10-01",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "copyright 2016-2017, 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": {
    "/oic/sec/pstat" : {
      "get": {
        "description": "This Resource specifies Device provisioning status.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"}
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example":
                {
                  "rt": ["oic.r.pstat"],
                  "dos": {"s": 3, "p": true},
                  "isop": true,
                  "cm": 8,
                  "tm": 60,
                  "om": 2,
                  "sm": 7,
                  "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
                },
              "schema": { "$ref": "#/definitions/Pstat" }
            },
            "400": {
              "description" : "The request is invalid."
            }
        }
      },
      "post": {
        "description": "Sets or updates Device provisioning status data.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": { "$ref": "#/definitions/Pstat-Update" },
            "x-example":
              {
                "dos": {"s": 3},
                "tm": 60,
                "om": 2,
                "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
              }
          }
        ],
        "responses": {
            "400": {
              "description" : "The request is invalid."
            },
            "204": {
              "description" : "The PSTAT entry is updated."
            }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : [ "oic.if.rw", "oic.if.baseline" ]
    }
  },
  "definitions": {
    "Pstat" : {
      "properties": {
        "rowneruuid": {
          "description": "The UUID formatted identity of the Resource owner\nFormat pattern
according to IETF RFC 4122.",
          "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}$",
          "type": "string"
        },
        "rt": {
          "description": "Resource Type of the Resource.",
          "items": {
            "maxLength": 64,
            "type": "string",
            "enum": ["oic.r.pstat"]
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "om": {
          "description": "Current operational mode\nDevice provisioning operation may be server
directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
and indicates the provisioning operation modes1 - Server-directed utilzing multiple provisioning
services2 - Server-directed utilzing a single provisioning service4 - Client-directed provisioning8
- Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
          "maximum": 7,
          "minimum": 1,
          "type": "integer"
        },
        "cm": {
          "description": "Current Device provisioning mode\nDevice provisioning mode maintains a
bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
- Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
Software Version Validation128 - Initiate Secure Software Update.",
          "maximum": 255,
          "minimum": 0,
          "type": "integer",
          "readOnly": true
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "isop": {
          "description": "true indicates Device is operational.",
          "readOnly": true,
          "type": "boolean"
        },
        "tm": {
          "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
- Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
Software Version Validation128 - Initiate Secure Software Update.",
          "maximum": 255,
          "minimum": 0,
          "type": "integer"
        },
        "sm": {
          "description": "Supported operational modes\nDevice provisioning operation may be server
directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
and indicates the provisioning operation modes1 - Server-directed utilzing multiple provisioning
services2 - Server-directed utilzing a single provisioning service4 - Client-directed provisioning8
- Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
          "maximum": 7,
          "minimum": 1,
          "type": "integer",
          "readOnly": true
        },
        "dos": {
          "description": "Device on-boarding state\nDevice operation state machine.",
          "properties": {
            "p": {
              "default": true,
              "description": "'p' is TRUE when the 's' state is pending until all necessary changes
to Device Resources are complete.",
              "readOnly": true,
              "type": "boolean"
            },
            "s": {
              "description": "The current or pending operational state.",
              "x-detail-desc": [
                "0 - RESET - Device reset state.",
                "1 - RFOTM - Ready for Device owner transfer method state.",
                "2 - RFPRO - Ready for Device provisioning state.",
                "3 - RFNOP - Ready for Device normal operation state.",
                "4 - SRESET - The Device is in a soft reset state."
              ],
              "maximum": 4,
              "minimum": 0,
              "type": "integer"
            }
          },
          "required": [
            "s"
          ],
          "type": "object"
        },
        "if" : {
          "description": "The interface set supported by this Resource.",
          "items": {
            "enum": [ "oic.if.rw", "oic.if.baseline" ],
            "type": "string"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        }
      },
      "type" : "object",
      "required": ["dos", "isop", "cm", "tm", "om", "sm", "rowneruuid"]
    },
    "Pstat-Update" : {
      "properties": {
        "rowneruuid": {
          "description": "The UUID formatted identity of the Resource owner\nFormat pattern
according to IETF RFC 4122.",
          "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}$",
          "type": "string"
        },
        "om": {
          "description": "Current operational mode\nDevice provisioning operation may be server
directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
and indicates the provisioning operation modes1 - Server-directed utilzing multiple provisioning
services2 - Server-directed utilzing a single provisioning service4 - Client-directed provisioning8
- Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
          "maximum": 7,
          "minimum": 1,
          "type": "integer"
        },
        "tm": {
          "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
- Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
Software Version Validation128 - Initiate Secure Software Update.",
          "maximum": 255,
          "minimum": 0,
          "type": "integer"
        },
        "dos": {
          "description": "Device on-boarding state\nDevice operation state machine.",
          "properties": {
            "p": {
              "default": true,
              "description": "'p' is TRUE when the 's' state is pending until all necessary changes
to Device Resources are complete.",
              "readOnly": true,
              "type": "boolean"
            },
            "s": {
              "description": "The current or pending operational state.",
              "x-detail-desc": [
                "0 - RESET - Device reset state.",
                "1 - RFOTM - Ready for Device owner transfer method state.",
                "2 - RFPRO - Ready for Device provisioning state.",
                "3 - RFNOP - Ready for Device normal operation state.",
                "4 - SRESET - The Device is in a soft reset state."
              ],
              "maximum": 4,
              "minimum": 0,
              "type": "integer"
            }
          },
          "required": [
            "s"
          ],
          "type": "object"
        }
      },
      "type" : "object"
    }
  }
```

```
}
```

C.6.5 Property definition

Table C-9 defines the Properties that are part of the "oic.r.pstat" Resource Type.

Table C-9 – The Property definitions of the Resource with type "rt" = "oic.r.pstat".

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	Yes	Read Write	The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
om	integer	Yes	Read Write	Current operational mode Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes1 - Server-directed utilzing multiple provisioning services2 - Server-directed utilzing a single provisioning service4 - Client-directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.
cm	integer	Yes	Read Only	Current Device provisioning mode Device provisioning mode maintains a bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 -

				Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
isop	boolean	Yes	Read Only	true indicates Device is operational.
tm	integer	Yes	Read Write	Target Device provisioning mode Device provisioning mode maintains a bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.
sm	integer	Yes	Read Only	Supported operational modes Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes1 - Server-directed utilzing multiple provisioning services2 - Server-directed utilzing a single provisioning service4 - Client-directed

				provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.
dos	object: see schema	Yes	Read Write	Device on-boarding state Device operation state machine.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
rowneruuid	string	No	Read Write	The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.
om	integer	No	Read Write	Current operational mode Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes1 - Server-directed utilzing multiple provisioning services2 - Server-directed utilzing a single provisioning service4 - Client-directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.
tm	integer	No	Read Write	Target Device provisioning mode Device provisioning mode maintains a bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management

				services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update.
dos	object: see schema	No	Read Write	Device on-boarding state Device operation state machine.

C.6.6 CRUDN behaviour

Table C-10 defines the CRUDN operations that are supported on the "oic.r.pstat" Resource Type.

Table C-10 – The CRUDN operations of the Resource with type "rt" = "oic.r.pstat".

Create	Read	Update	Delete	Notify
	get	post		observe

C.7 Asserted Roles

C.7.1 Introduction

This Resource specifies roles that have been asserted.

C.7.2 Well-known URI

/oic/sec/roles

C.7.3 Resource type

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

C.7.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Asserted Roles",
    "version": "2017-03-23",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "copyright 2016-2017, 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": {
    "/oic/sec/roles" : {
      "get": {
        "description": "This Resource specifies roles that have been asserted.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"}
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example":
                {
                  "roles" :[
                      {
                        "credid":1,
                        "credtype":8,
                        "subjectuuid":"00000000-0000-0000-0000-000000000000",
                        "publicdata":
                          {
                             "encoding":"oic.sec.encoding.pem",
                             "data":"PEMENCODEDROLECERT"
                          },
                        "optionaldata":
                          {
                             "revstat": false,
                             "encoding":"oic.sec.encoding.pem",
                             "data":"PEMENCODEDISSUERCERT"
                          }
                      },
                      {
                        "credid":2,
                        "credtype":8,
                        "subjectuuid":"00000000-0000-0000-0000-000000000000",
                        "publicdata":
                          {
                             "encoding":"oic.sec.encoding.pem",
                             "data":"PEMENCODEDROLECERT"
                          },
                        "optionaldata":
                          {
                             "revstat": false,
                             "encoding":"oic.sec.encoding.pem",
                             "data":"PEMENCODEDISSUERCERT"
                          }
                      }
                  ],
                  "rt":["oic.r.roles"],
                  "if":[ "oic.if.rw" ]
                }
                ,
              "schema": { "$ref": "#/definitions/Roles" }
            },
            "400": {
              "description" : "The request is invalid."
            }
        }
      },
      "post": {
        "description": "Update the roles Resource, i.e., assert new roles to this server.\n\nNew
role certificates that match an existing certificate (i.e., publicdata\nand optionaldata are the
same) are not added to the Resource (and 204 is\nreturned).\n\nThe provided credid values are
ignored, the Resource assigns its own.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": { "$ref": "#/definitions/Roles-update" },
            "x-example":
              {
                "roles" :[
                    {
                      "credid":1,
                      "credtype":8,
                      "subjectuuid":"00000000-0000-0000-0000-000000000000",
                      "publicdata":
                        {
                           "encoding":"oic.sec.encoding.pem",
                           "data":"PEMENCODEDROLECERT"
                        },
                      "optionaldata":
                        {
                           "revstat": false,
                           "encoding":"oic.sec.encoding.pem",
                           "data":"PEMENCODEDISSUERCERT"
                        }
                    },
                    {
                      "credid":2,
                      "credtype":8,
                      "subjectuuid":"00000000-0000-0000-0000-000000000000",
                      "publicdata":
                        {
                           "encoding":"oic.sec.encoding.pem",
                           "data":"PEMENCODEDROLECERT"
                        },
                      "optionaldata":
                        {
                           "revstat": false,
                           "encoding":"oic.sec.encoding.pem",
                           "data":"PEMENCODEDISSUERCERT"
                        }
                    }
                ]
              }
          }
        ],
        "responses": {
            "400": {
              "description" : "The request is invalid."
            },
            "204": {
              "description" : "The roles entry is updated."
            }
        }
      },
      "delete": {
        "description": "Deletes roles Resource entries.\nWhen DELETE is used without query
parameters, all the roles entries are deleted.\nWhen DELETE is used with a query parameter, only the
entries matching\nthe query parameter are deleted.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {"$ref": "#/parameters/roles-filtered"}
        ],
        "responses": {
            "200": {
              "description" : "The specified or all roles Resource entries have been successfully
deleted."
            },
            "400": {
              "description" : "The request is invalid."
            }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : [ "oic.if.rw", "oic.if.baseline" ]
    },
    "roles-filtered" : {
      "in" : "query",
      "name" : "credid",
      "required" : false,
      "type" : "integer",
      "description" : "Only applies to the credential with the specified credid.",
      "x-example" : 2112
    }
  },
  "definitions": {
    "Roles" : {
      "properties": {
        "rt": {
          "description": "Resource Type of the Resource.",
          "items": {
            "maxLength": 64,
            "type": "string",
            "enum": ["oic.r.roles"]
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "roles": {
          "description": "List of role certificates.",
          "items": {
            "properties": {
              "credid": {
                "description": "Local reference to a credential Resource.",
                "type": "integer"
              },
              "credtype": {
                "description": "Representation of this credential's type\nCredential Types - Cred
type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
password32 - Asymmetric encryption key.",
                "maximum": 63,
                "minimum": 0,
                "type": "integer"
              },
              "credusage": {
                "description": "A string that provides hints about how/where the cred is used\nThe
type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
                "enum": [
                  "oic.sec.cred.trustca",
                  "oic.sec.cred.cert",
                  "oic.sec.cred.rolecert",
                  "oic.sec.cred.mfgtrustca",
                  "oic.sec.cred.mfgcert"
                ],
                "type": "string"
              },
              "crms": {
                "description": "The refresh methods that may be used to update this credential.",
                "items": {
                  "description": "Each enum represents a method by which the credentials are
refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
                  "enum": [
                    "oic.sec.crm.pro",
                    "oic.sec.crm.psk",
                    "oic.sec.crm.rdp",
                    "oic.sec.crm.skdc",
                    "oic.sec.crm.pk10"
                  ],
                  "type": "string"
                },
                "type": "array"
              },
              "optionaldata": {
                "description": "Credential revocation status information\nOptional credential
contents describes revocation status for this credential.",
                "properties": {
                  "data": {
                    "description": "This is the encoded structure.",
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the optdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
                      "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
                      "oic.sec.encoding.base64 - Base64 encoded object.",
                      "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
                      "oic.sec.encoding.der - Encoding for DER encoded certificate.",
                      "oic.sec.encoding.raw - Raw hex encoded data."
                    ],
                    "enum": [
                      "oic.sec.encoding.jwt",
                      "oic.sec.encoding.cwt",
                      "oic.sec.encoding.base64",
                      "oic.sec.encoding.pem",
                      "oic.sec.encoding.der",
                      "oic.sec.encoding.raw"
                    ],
                    "type": "string"
                  },
                  "revstat": {
                    "description": "Revocation status flag - true = revoked.",
                    "type": "boolean"
                  }
                },
                "required": [
                  "revstat"
                ],
                "type": "object"
              },
              "period": {
                "description": "String with RFC5545 Period.",
                "type": "string"
              },
              "privatedata": {
                "description": "Private credential information\nCredential Resource non-public
contents.",
                "properties": {
                  "data": {
                    "description": "The encoded value.",
                    "maxLength": 3072,
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the privdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
                      "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
                      "oic.sec.encoding.base64 - Base64 encoded object.",
                      "oic.sec.encoding.uri - URI reference.",
                      "oic.sec.encoding.handle - Data is contained in a storage sub-system
referenced using a handle.",
                      "oic.sec.encoding.raw - Raw hex encoded data."
                    ],
                    "enum": [
                      "oic.sec.encoding.jwt",
                      "oic.sec.encoding.cwt",
                      "oic.sec.encoding.base64",
                      "oic.sec.encoding.uri",
                      "oic.sec.encoding.handle",
                      "oic.sec.encoding.raw"
                    ],
                    "type": "string"
                  },
                  "handle": {
                    "description": "Handle to a key storage Resource.",
                    "type": "integer"
                  }
                },
                "required": [
                  "encoding"
                ],
                "type": "object"
              },
              "publicdata": {
                "description": "Public credential information.",
                "properties": {
                  "data": {
                    "description": "This is the encoded value.",
                    "maxLength": 3072,
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the pubdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
                      "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
                      "oic.sec.encoding.base64 - Base64 encoded object.",
                      "oic.sec.encoding.uri - URI reference.",
                      "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
                      "oic.sec.encoding.der - Encoding for DER encoded certificate.",
                      "oic.sec.encoding.raw - Raw hex encoded data."
                    ],
                    "enum": [
                      "oic.sec.encoding.jwt",
                      "oic.sec.encoding.cwt",
                      "oic.sec.encoding.base64",
                      "oic.sec.encoding.uri",
                      "oic.sec.encoding.pem",
                      "oic.sec.encoding.der",
                      "oic.sec.encoding.raw"
                    ],
                    "type": "string"
                  }
                },
                "type": "object"
              },
              "roleid": {
                "description": "The role this credential possesses\nSecurity role specified as an
<Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
                "properties": {
                  "authority": {
                    "description": "The Authority component of the entity being identified. A NULL
<Authority> refers to the local entity or Device.",
                    "type": "string"
                  },
                  "role": {
                    "description": "The ID of the role being identified.",
                    "type": "string"
                  }
                },
                "required": [
                  "role"
                ],
                "type": "object"
              },
              "subjectuuid": {
                "anyOf": [
                  {
                    "description": "The id of the Device, which the cred entry applies to or \"*\"
for wildcard identity.",
                    "pattern": "^\\*$",
                    "type": "string"
                  },
                  {
                    "description": "Format pattern according to IETF RFC 4122.",
                    "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
                    "type": "string"
                  }
                ]
              }
            },
            "type": "object"
          },
          "type": "array"
        },
        "if": {
          "description": "The interface set supported by this Resource.",
          "items": {
            "enum": [ "oic.if.rw", "oic.if.baseline" ],
            "type": "string"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        }
      },
      "type" : "object",
      "required": ["roles"]
    },
    "Roles-update" : {
      "properties": {
        "roles": {
          "description": "List of role certificates.",
          "items": {
            "properties": {
              "credid": {
                "description": "Local reference to a credential Resource.",
                "type": "integer"
              },
              "credtype": {
                "description": "Representation of this credential's type\nCredential Types - Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 - Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or password32 - Asymmetric encryption key.",
                "maximum": 63,
                "minimum": 0,
                "type": "integer"
              },
              "credusage": {
                "description": "A string that provides hints about how/where the cred is used\nThe type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert - Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
                "enum": [
                  "oic.sec.cred.trustca",
                  "oic.sec.cred.cert",
                  "oic.sec.cred.rolecert",
                  "oic.sec.cred.mfgtrustca",
                  "oic.sec.cred.mfgcert"
                ],
                "type": "string"
              },
              "crms": {
                "description": "The refresh methods that may be used to update this credential.",
                "items": {
                  "description": "Each enum represents a method by which the credentials are refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp - Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
                  "enum": [
                    "oic.sec.crm.pro",
                    "oic.sec.crm.psk",
                    "oic.sec.crm.rdp",
                    "oic.sec.crm.skdc",
                    "oic.sec.crm.pk10"
                  ],
                  "type": "string"
                },
                "type": "array"
              },
              "optionaldata": {
                "description": "Credential revocation status information\nOptional credential
contents describes revocation status for this credential.",
                "properties": {
                  "data": {
                    "description": "This is the encoded structure.",
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the optdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
                      "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
                      "oic.sec.encoding.base64 - Base64 encoded object.",
                      "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
                      "oic.sec.encoding.der - Encoding for DER encoded certificate.",
                      "oic.sec.encoding.raw - Raw hex encoded data."
                    ],
                    "enum": [
                      "oic.sec.encoding.jwt",
                      "oic.sec.encoding.cwt",
                      "oic.sec.encoding.base64",
                      "oic.sec.encoding.pem",
                      "oic.sec.encoding.der",
                      "oic.sec.encoding.raw"
                    ],
                    "type": "string"
                  },
                  "revstat": {
                    "description": "Revocation status flag - true = revoked.",
                    "type": "boolean"
                  }
                },
                "required": [
                  "revstat"
                ],
                "type": "object"
              },
              "period": {
                "description": "String with RFC5545 Period.",
                "type": "string"
              },
              "privatedata": {
                "description": "Private credential information\nCredential Resource non-public
contents.",
                "properties": {
                  "data": {
                    "description": "The encoded value.",
                    "maxLength": 3072,
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the privdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
                      "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
                      "oic.sec.encoding.base64 - Base64 encoded object.",
                      "oic.sec.encoding.uri - URI reference.",
                      "oic.sec.encoding.handle - Data is contained in a storage sub-system
referenced using a handle.",
                      "oic.sec.encoding.raw - Raw hex encoded data."
                    ],
                    "enum": [
                      "oic.sec.encoding.jwt",
                      "oic.sec.encoding.cwt",
                      "oic.sec.encoding.base64",
                      "oic.sec.encoding.uri",
                      "oic.sec.encoding.handle",
                      "oic.sec.encoding.raw"
                    ],
                    "type": "string"
                  },
                  "handle": {
                    "description": "Handle to a key storage Resource.",
                    "type": "integer"
                  }
                },
                "required": [
                  "encoding"
                ],
                "type": "object"
              },
              "publicdata": {
                "description": "Public credential information.",
                "properties": {
                  "data": {
                    "description": "The encoded value.",
                    "maxLength": 3072,
                    "type": "string"
                  },
                  "encoding": {
                    "description": "A string specifying the encoding format of the data contained in
the pubdata.",
                    "x-detail-desc": [
                      "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
                      "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
                      "oic.sec.encoding.base64 - Base64 encoded object.",
                      "oic.sec.encoding.uri - URI reference.",
                      "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
                      "oic.sec.encoding.der - Encoding for DER encoded certificate.",
                      "oic.sec.encoding.raw - Raw hex encoded data."
                    ],
                    "enum": [
                      "oic.sec.encoding.jwt",
                      "oic.sec.encoding.cwt",
                      "oic.sec.encoding.base64",
                      "oic.sec.encoding.uri",
                      "oic.sec.encoding.pem",
                      "oic.sec.encoding.der",
                      "oic.sec.encoding.raw"
                    ],
                    "type": "string"
                  }
                },
                "type": "object"
              },
              "roleid": {
                "description": "The role this credential possesses\nSecurity role specified as an
<Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
                "properties": {
                  "authority": {
                    "description": "The Authority component of the entity being identified. A NULL
<Authority> refers to the local entity or Device.",
                    "type": "string"
                  },
                  "role": {
                    "description": "The ID of the role being identified.",
                    "type": "string"
                  }
                },
                "required": [
                  "role"
```

```
        ],
        "type": "object"
       },
       "subjectuuid": {
        "anyOf": [
         {
          "description": "The id of the Device, which the cred entry applies to or \"*\"
for wildcard identity.",
          "pattern": "^\\*$",
          "type": "string"
         },
         {
          "description": "Format pattern according to IETF RFC 4122.",
          "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
F0-9]{12}$",
          "type": "string"
         }
        ]
       }
      },
      "type": "object"
     },
     "type": "array"
    }
   },
   "type" : "object",
   "required": ["roles"]
  }
 }
}
```

C.7.5 Property definition

Table C-11 defines the Properties that are part of the "oic.r.roles" Resource Type.

Table C-11 – The Property definitions of the Resource with type "rt" = "oic.r.roles".

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
roles	array: see schema	Yes	Read Write	List of role certificates.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
roles	array: see schema	Yes	Read Write	List of role certificates.

C.7.6 CRUDN behaviour

Table C-12 defines the CRUDN operations that are supported on the "oic.r.roles" Resource Type.

Table C-12 – The CRUDN operations of the Resource with type "rt" = "oic.r.roles".

Create	Read	Update	Delete	Notify
	get	post	delete	observe

C.8 Security Profile

C.8.1 Introduction

Resource specifying supported and active security profile(s).

C.8.2 Well-known URI

/oic/sec/sp

C.8.3 Resource type

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

C.8.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Security Profile",
    "version": "2019-02-08",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "copyright 2016-2017, 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": {
    "/oic/sec/sp" : {
      "get": {
        "description": "Resource specifying supported and active security profile(s).\n",
        "parameters": [
          {"$ref": "#/parameters/interface"}
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example":
                {
                  "rt": ["oic.r.sp"],
                  "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
                  "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
                },
              "schema": { "$ref": "#/definitions/SP" }
            },
            "400": {
              "description" : "The request is invalid."
            }
        }
      },
      "post": {
        "description": "Sets or updates Device provisioning status data.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": { "$ref": "#/definitions/SP-Update" },
            "x-example":
              {
                "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
                "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
              }
          }
        ],
        "responses": {
            "200": {
              "description" : "",
              "x-example":
                {
                  "rt": ["oic.r.sp"],
                  "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
                  "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
                },
              "schema": { "$ref": "#/definitions/SP" }
            },
            "400": {
              "description" : "The request is invalid."
            }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : [ "oic.if.rw", "oic.if.baseline" ]
    }
  },
  "definitions": {
    "SP" : {
      "properties": {
        "rt": {
          "description": "Resource Type of the Resource.",
          "items": {
            "maxLength": 64,
            "type": "string",
            "enum": ["oic.r.sp"]
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "currentprofile": {
          "description": "Security Profile currently active.",
          "type": "string"
        },
        "supportedprofiles": {
          "description": "Array of supported Security Profiles.",
          "items": {
            "type": "string"
          },
          "type": "array"
        },
        "if": {
          "description": "The interface set supported by this Resource.",
          "items": {
            "enum": [ "oic.if.rw", "oic.if.baseline" ],
            "type": "string"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        }
      },
      "type" : "object",
      "required": ["supportedprofiles", "currentprofile"]
    },
    "SP-Update" : {
      "properties": {
        "currentprofile": {
          "description": "Security Profile currently active.",
          "type": "string"
        },
        "supportedprofiles": {
          "description": "Array of supported Security Profiles.",
          "items": {
            "type": "string"
          },
          "type": "array"
        }
      },
      "type" : "object"
    }
  }
}
```

C.8.5 Property definition

Table C-13 defines the Properties that are part of the "oic.r.sp" Resource Type.

Table C-13 – The Property definitions of the Resource with type "rt" = "oic.r.sp".

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
currentprofile	string	Yes	Read Write	Security Profile currently active.
supportedprofiles	array: see schema	Yes	Read Write	Array of supported Security Profiles.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
currentprofile	string		Read Write	Security Profile currently active.
supportedprofiles	array: see schema		Read Write	Array of supported Security Profiles.

C.8.6 CRUDN behaviour

Table C-14 defines the CRUDN operations that are supported on the "oic.r.sp" Resource Type.

Table C-14 – The CRUDN operations of the Resource with type "rt" = "oic.r.sp".

Create	Read	Update	Delete	Notify
	get	post		observe

C.9 Auditable Event List

C.9.1 Introduction

This Resource contains the Auditable Events that have been logged on the Device.

C.9.2 Well-known URI

/oic/sec/ael

C.9.3 Resource type

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

C.9.4 OpenAPI 2.0 definition

```
{
        "swagger": "2.0",
        "info": {
                "title": "Auditable Event List",
                "version": "2019-10-03",
                "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": {
                "/AelResURI": {
                        "get": {
                                "description": "This Resource contains the Auditable Events that have
been logged on the Device.",
                                "parameters": [{"$ref": "#/parameters/interface"}],
                                "responses": {
                                        "200": {
                                                "description": "Example response payload. In this
example, 'oic.d.light' Device has logged 2 Auditable Event Entries: Update attempt against
'/room1/led1' Resource was denied, and Delete attempt against '/room1/led1' Resource was denied.
Both Auditable Event Entries belong to 'AccessControl (0x01)' category and 'WARN' priority (2).",
                                                "x-example": {
                                                        "rt": [ "oic.r.ael" ],
                                                        "events": [
                                                                {
                                                                        "aeid": "AC-1",
                                                                        "category": 1,
                                                                        "priority": 2,
                                                                        "timestamp": "2018-11-
13T20:22:39+00:00",
                                                                        "message":  "Access Denied",
                                                                        "auxiliaryinfo":
[ "[2001::1]:1234", "0f33887b-f7d6-4fdb-9125-dd4b60d5aaae", "/room1/led1", "UPDATE", "RFNOP", "No
roles asserted" ]
                                                                },
                                                                {
                                                                        "aeid": "AC-1",
                                                                        "category": 1,
                                                                        "priority": 2,
                                                                        "timestamp": "2018-11-
13T20:20:00+00:00",
                                                                        "message":  "Access Denied",
                                                                        "auxiliaryinfo":
[ "[2001::1]:1234", "0f33887b-f7d6-4fdb-9125-dd4b60d5aaae", "/room1/led1", "DELETE", "RFNOP", "No
roles asserted" ]
                                                                }
```

```
                            ],
                            "usedspace": 2,
                            "maxspace": 5,
                            "categoryfilter": 3,
                            "priorityfilter": 1
                        },
                        "schema": { "$ref": "#/definitions/Ael" }
                    }
                }
            },
            "post": {
                "description": "An UPDATE operation may set the 'categoryfilter'
and/or 'priorityfilter' Properties.",
                "parameters": [
                    {
                        "$ref": "#/parameters/interface"
                    },
                    {
                        "in": "body",
                        "name": "body",
                        "required": true,
                        "schema": { "$ref": "#/definitions/Ael-Update" },
                        "x-example": {
                            "categoryfilter": 3,
                            "priorityfilter": 1
                        }
                    }
                ],
                "responses": {
                    "204": {
                        "description": "The new categoryfilter and
priorityfilter were set."
                    }
                }
            }
        }
    },
    "parameters": {
        "interface": {
            "in": "query",
            "name": "if",
            "type": "string",
            "enum": [ "oic.if.rw", "oic.if.baseline" ]
        }
    },
    "definitions": {
        "Aee": {
            "description": "Auditable Event Entry logged by a Device",
            "type": "object",
            "properties": {
                "aeid": {
                    "description": "Identity of the logged event",
                    "type": "string",
                    "readOnly": true
                },
                "category" : {
                    "description": "Category of this Auditable Event: 0x01
(Access Control), 0x02 (Onboarding), 0x04 (Device), 0x08 (Authentication), 0x10 (SVR Modification),
0x20 (Cloud), 0x40 (Communication), 0x80 (Reserved)",
                    "type": "integer",
                    "enum": [
                        1, 2, 4, 8, 16, 32, 64, 128
                    ],
                    "readOnly": true
                },
                "priority": {
                    "definitions": "Priority of this Auditable Event: 0 (CRIT), 1
(ERR), 2 (WARN), 3 (INFO), 4 (DEBUG)",
                    "type": "integer",
                    "enum": [
                        0, 1, 2, 3, 4
                    ],
                                        "readOnly": true
                                },
                                "timestamp": {
                                        "description": "Time when this Auditable Event occured",
                                        "type": "string",
                                        "format": "date-time",
                                        "readOnly": true
                                },
                                "message": {
                                        "description": "Description for this Auditable Event",
                                        "type": "string",
                                        "readOnly": true
                                },
                                "auxiliaryinfo": {
                                        "description": "Supplementary info for Auditable Event
message. (e.g. URI of specific Resource in ACE2 for 'Access Denied' message)",
                                        "type": "array",
                                        "minItems": 0,
                                        "items": {
                                                "type": "string"
                                        },
                                        "readOnly": true
                                }
                        },
                        "required": [
                                "aeid", "message", "auxiliaryinfo", "category", "priority",
"timestamp"
                        ]
                },

                "Ael": {
                        "description": "Resource for storing Auditable Events List",
                        "type": "object",
                        "properties": {
                                "rt": {
                                        "description": "Resource Type",
                                        "type": "array",
                                        "minItems": 1,
                                        "uniqueItems": true,
                                        "items": {
                                                "maxLength": 64,
                                                "type": "string",
                                                "enum": [ "oic.r.ael" ]
                                        },
                                        "readOnly": true
                                },
                                "n": {
                                        "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
                                },
                                "id": {
                                        "$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",
                                        "type": "array",
                                        "minItems": 2,
                                        "uniqueItems": true,
                                        "items": {
                                                "type": "string",
                                                "enum": [ "oic.if.rw", "oic.if.baseline" ]
                                        },
                                        "readOnly": true
                                },
                                "events": {
                                        "description": "This list stores AEEs whose 'category'
Property value is filtered by 'categoryfilter' Property and 'priority' Property value is equal or
less than the value of 'priorityfilter' Property.",
                    "type": "array",
                    "uniqueItems": true,
                    "items": {
                        "$ref": "#/definitions/Aee"
                    }
                },
                "usedspace": {
                    "description": "Current used space for logged AEEs. The
Device updates this Property whenever new AEEs are logged.",
                    "type": "integer",
                    "default": 0,
                    "readOnly": true
                },
                "maxspace": {
                    "description": "This means the maximum allowable storage size
for AEEs that can be stored in 'events' list. The Manufacturer chooses this value.",
                    "type": "integer",
                    "readOnly": true
                },
                "unit":  {
                    "description": "The unit for 'usedspace' and 'maxspace'
Properties. The Manufacturer chooses this value.",
                    "type": "string",
                    "enum": [
                        "Kbyte",
                        "Byte"
                    ],
                    "default": "Byte",
                    "readOnly": true
                },
                "categoryfilter": {
                    "description" : "This value decides what categories of AEEs
are to be logged. Meaning of each bit: 0x01 (Access Control), 0x02 (Onboarding), 0x04 (Device), 0x08
(Authentication), 0x10 (SVR Modification), 0x20 (Cloud), 0x40 (Communication), 0x80 (Reserved).
e.g.) if categoryfilter == 0xff: log all events of all categories,  e.g.) if categoryfilter == 0x03:
log all events of 'AC (== 0x01)' and 'OB (==0x02)' categories ",
                    "type": "integer",
                    "default": 255
                },
                "priorityfilter": {
                    "description": "The AEEs whose 'priority' values are equal to
or smaller than this value are logged. A smaller value means a higher priority. Meaning of each
value: 0 (CRIT), 1 (ERR), 2 (WARN), 3 (INFO), 4 (DEBUG). e.g.) if priorityfilter is set to DEBUG
(==4) all AEEs will be logged, e.g.) if priorityfilter is set to 1, CRIT (==0) and ERR (==1) AEEs
will be logged ",
                    "type": "integer",
                    "default": 4,
                    "enum": [
                        0, 1, 2, 3, 4
                    ]
                }
            },
            "required": [
                "events", "usedspace", "maxspace", "categoryfilter", "priorityfilter"
            ]
        },
        "Ael-Update": {
            "type": "object",
            "properties": {
                "categoryfilter": {
                    "description" : "This value decides what categories of AEEs
are to be logged. Meaning of each bit: 0x01 (Access Control), 0x02 (Onboarding), 0x04 (Device), 0x08
(Authentication), 0x10 (SVR Modification), 0x20 (Cloud), 0x40 (Communication). e.g.) if
categoryfilter == 0xff: log all events of all categories,  e.g.) if categoryfilter == 0x03: log all
events of 'AC (== 0x01)' and 'OB (==0x02)' categories ",
                    "type": "integer",
                    "default": 255
                },
                "priorityfilter": {
                    "description": "The AEEs whose 'priority' values are equal to
or smaller than this value are logged. A smaller value means a higher priority. Meaning of each
value: 0 (CRIT), 1 (ERR), 2 (WARN), 3 (INFO), 4 (DEBUG). e.g.) if priorityfilter is set to DEBUG
```

```
(==4) all AEEs will be logged, e.g.) if priorityfilter is set to 1, CRIT (==0) and ERR (==1) AEEs
will be logged ",
                                        "type": "integer",
                                        "default": 4,
                                        "enum": [
                                                0, 1, 2, 3, 4
                                        ]
                                }
                        },
                        "required": [
                                "categoryfilter", "priorityfilter"
                        ]

                }
        }
}
```

C.9.5 Property definition

Table C-15 defines the Properties that are part of the "oic.r.ael" Resource Type.

Table C-15 – The Property definitions of the Resource with type "rt" = "oic.r.ael".

Property name	Value type	Mandatory	Access mode	Description
aeid	string	Yes	Read Only	Identity of the logged event
category	integer	Yes	Read Only	Category of this Auditable Event: 0x01 (Access Control), 0x02 (Onboarding), 0x04 (Device), 0x08 (Authentication), 0x10 (SVR Modification), 0x20 (Cloud), 0x40 (Communication), 0x80 (Reserved)
priority	integer	Yes	Read Only	
timestamp	string	Yes	Read Only	Time when this Auditable Event occured
message	string	Yes	Read Only	Description for this Auditable Event
auxiliaryinfo	array: see schema	Yes	Read Only	Supplementary info for Auditable Event message. (e.g. URI of specific Resource in ACE2 for 'Access Denied' message)
rt	array: see schema	No	Read Only	Resource Type
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The OCF Interface set supported by this Resource
events	array: see schema	Yes	Read Write	This list stores AEEs whose 'category' Property value is filtered by

				'categoryfilter' Property and 'priority' Property value is equal or less than the value of 'priorityfilter' Property.
usedspace	integer	Yes	Read Only	Current used space for logged AEEs. The Device updates this Property whenever new AEEs are logged.
maxspace	integer	Yes	Read Only	This means the maximum allowable storage size for AEEs that can be stored in 'events' list. The Manufacturer chooses this value.
unit	string	No	Read Only	The unit for 'usedspace' and 'maxspace' Properties. The Manufacturer chooses this value.
categoryfilter	integer	Yes	Read Write	This value decides what categories of AEEs are to be logged. Meaning of each bit: 0x01 (Access Control), 0x02 (Onboarding), 0x04 (Device), 0x08 (Authentication), 0x10 (SVR Modification), 0x20 (Cloud), 0x40 (Communication), 0x80 (Reserved). e.g.) if categoryfilter == 0xff: log all events of all categories, e.g.) if categoryfilter == 0x03: log all events of 'AC (== 0x01)' and 'OB (==0x02)' categories
priorityfilter	integer	Yes	Read Write	The AEEs whose 'priority' values are equal to or smaller than this value are logged. A smaller value means a higher priority. Meaning of each value: 0 (CRIT), 1 (ERR), 2 (WARN), 3 (INFO), 4 (DEBUG). e.g.) if priorityfilter is set to DEBUG (==4) all AEEs will be logged, e.g.) if priorityfilter is set to 1, CRIT (==0) and

				ERR (==1) AEEs will be logged
categoryfilter	integer	Yes	Read Write	This value decides what categories of AEEs are to be logged. Meaning of each bit: 0x01 (Access Control), 0x02 (Onboarding), 0x04 (Device), 0x08 (Authentication), 0x10 (SVR Modification), 0x20 (Cloud), 0x40 (Communication). e.g.) if categoryfilter == 0xff: log all events of all categories, e.g.) if categoryfilter == 0x03: log all events of 'AC (== 0x01)' and 'OB (==0x02)' categories
priorityfilter	integer	Yes	Read Write	The AEEs whose 'priority' values are equal to or smaller than this value are logged. A smaller value means a higher priority. Meaning of each value: 0 (CRIT), 1 (ERR), 2 (WARN), 3 (INFO), 4 (DEBUG). e.g.) if priorityfilter is set to DEBUG (==4) all AEEs will be logged, e.g.) if priorityfilter is set to 1, CRIT (==0) and ERR (==1) AEEs will be logged

C.9.6 CRUDN behaviour

Table C-16 defines the CRUDN operations that are supported on the "oic.r.ael" Resource Type.

Table C-16 – The CRUDN operations of the Resource with type "rt" = "oic.r.ael".

Create	Read	Update	Delete	Notify
	get	post		observe

C.10 Security Domain Information

C.10.1 Introduction

This Resource contains the information that identifies the OCF Security Domain to which the device belongs.

C.10.2 Well-known URI

/oic/sec/sdi

C.10.3 Resource type

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

C.10.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Security Domain Information",
    "version": "2019-10-01",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "copyright 2016-2017, 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": {
    "/oic/sec/sdi" : {
      "get": {
        "description": "This Resource contains the information that identifies the OCF Security
Domain to which the device belongs.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"}
        ],
        "responses": {
            "200": {
              "description" : "Success",
              "x-example":
                {
                  "rt": ["oic.r.sdi"],
                  "uuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
                  "name": "Home",
                  "priv": true
                },
              "schema": { "$ref": "#/definitions/Sdi" }
            },
            "400": {
              "description" : "The request is invalid."
            }
        }
      },
      "post": {
        "description": "Provision the OCF Security Domain information.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"},
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": { "$ref": "#/definitions/Sdi-Update" },
            "x-example": {
              "uuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
              "name": "Home",
              "priv": false
            }
          }
        ],
        "responses": {
            "400": {
              "description" : "The request is invalid."
            },
            "204": {
              "description" : "The SDI is updated.",
              "schema": { "$ref": "#/definitions/Sdi-Update" },
              "x-example": {
                "uuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
                "name": "Home",
                "priv": false
              }
            }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : [ "oic.if.rw", "oic.if.baseline" ]
    }
  },
  "definitions": {
    "Sdi" : {
      "properties": {
        "uuid": {
          "$ref": "https://openconnectivityfoundation.github.io/core/schemas/oic.types-
schema.json#/definitions/uuid"
        },
        "name": {
          "description": "Human-friendly name for the Security Domain, set by DOTS during
onboarding.",
          "type": "string"
        },
        "rt": {
          "description": "Resource Type of the Resource.",
          "items": {
            "maxLength": 64,
            "type": "string",
            "enum": ["oic.r.sdi"]
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        },
        "n": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/n"
        },
        "id": {
          "$ref":
"https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
schema.json#/definitions/id"
        },
        "priv": {
          "description": "Flag to indicate whether the Security Domain Information is copied to
“/oic/res”, and thus, whether it is publicly visible or private.",
          "type": "boolean"
        },
        "if" : {
          "description": "The interface set supported by this Resource.",
          "items": {
            "enum": [ "oic.if.rw", "oic.if.baseline" ],
            "type": "string"
          },
          "minItems": 1,
          "readOnly": true,
          "type": "array"
        }
      },
      "type" : "object",
      "required": [ "uuid", "name", "priv" ]
    },

    "Sdi-Update" : {
      "properties": {
        "uuid": {
          "$ref": "https://openconnectivityfoundation.github.io/core/schemas/oic.types-
schema.json#/definitions/uuid"
        },
        "name": {
          "description": "Human-friendly name for the Security Domain, set by DOTS during
onboarding.",
          "type": "string"
        },
        "priv": {
          "description": "Flag to indicate whether the Security Domain Information is copied to
“/oic/res”, and thus, whether it is publicly visible or private.",
          "type": "boolean"
        }
      },
      "type" : "object",
      "required": [ "name", "priv" ]
    }
  }
}
```

C.10.5 Property definition

Table C-17 defines the Properties that are part of the "oic.r.sdi" Resource Type.

Table C-17 – The Property definitions of the Resource with type "rt" = "oic.r.sdi".

Property name	Value type	Mandatory	Access mode	Description
uuid	multiple types: see schema	Yes	Read Write	
name	string	Yes	Read Write	Human-friendly name for the Security Domain, set by DOTS during onboarding.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
priv	boolean	Yes	Read Write	Flag to indicate whether the Security Domain Information is copied to “/oic/res”, and thus, whether it is publicly visible or private.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
uuid	multiple types: see schema	No	Read Write	
name	string	Yes	Read Write	Human-friendly name for the Security Domain, set by DOTS during onboarding.
priv	boolean	Yes	Read Write	Flag to indicate whether the Security Domain Information is copied to

				“/oic/res”, and thus, whether it is publicly visible or private.

C.10.6 CRUDN behaviour

Table C-18 defines the CRUDN operations that are supported on the "oic.r.sdi" Resource Type.

Table C-18 – The CRUDN operations of the Resource with type "rt" = "oic.r.sdi".

Create	Read	Update	Delete	Notify
	get	post		observe

Annex D
(informative)

OID definitions

This annex captures the OIDs defined throughout the document. The OIDs listed are intended to be used within the context of an X.509 v3 certificate. MAX is an upper bound for SEQUENCES of UTF8Strings and OBJECT IDENTIFIERs and should not exceed 255.

```
id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
     private(4) enterprise(1) OCF(51414) }

-- OCF Security specific OIDs

id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }
id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }

-- OCF Security Categories

id-ocfSecurityProfile ::= { id-ocfSecurity 0 }
id-ocfCertificatePolicy ::= { id-ocfSecurity 1 }

-- OCF Security Profiles

sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }

sp-unspecified-v0 ::= ocfSecurityProfileOID (id-sp-unspecified 0}
sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}

ocfSecurityProfileOID ::= UTF8String

-- OCF Security Certificate Policies

ocfCertificatePolicy-v1 ::= { id-ocfCertificatePolicy 2}

-- OCF X.509v3 Extensions

id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }

ocfVersion ::= SEQUENCE {
     major    INTEGER,
     minor    INTEGER,
     build    INTEGER}

ocfCompliance ::= SEQUENCE {
     version         ocfVersion,
     securityProfile SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
     deviceName      UTF8String,
     deviceManufacturer    UTF8String}

claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
```

```
ocfSecurityClaimsOID ::= OBJECT IDENTIFIER

ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID

cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }

ocfCPLAttributes ::= SEQUENCE {
     cpl-at-IANAPen UTF8String,
     cpl-at-model   UTF8String,
     cpl-at-version UTF8String}
```

Annex E
(informative)

Security considerations specific to Bridged Protocols

The text in this Annex is provided for information only. This Annex has no normative impact. This information is applicable at the time of initial publication and may become out of date.

E.1 Security Considerations specific to the AllJoyn Protocol

This clause intentionally left empty.

E.2 Security Considerations specific to the Bluetooth LE Protocol

BLE GAP supports two security modes, security mode 1 and security mode 2. Each security mode has several security levels (see Table E.1)

Security mode 1 and Security level 2 or higher would typically be considered secure from an OCF perspective. The appropriate selection of security mode and level is left to the vendor.

Table E.1 GAP security mode

GAP security mode	security level
Security mode 1	1 (no security)
	2 (Unauthenticated pairing with encryption)
	3 (Authenticated pairing with encryption)
	4 (Authenticated LE Secure Connections pairing with encryption)
Security mode 2	1 (Unauthenticated pairing with data signing)
	2 (Authenticated pairing with data signing)

Figure E-1 shows how communications in both ecosystems of OCF-BLE Bridge Platform are secured by their own security.

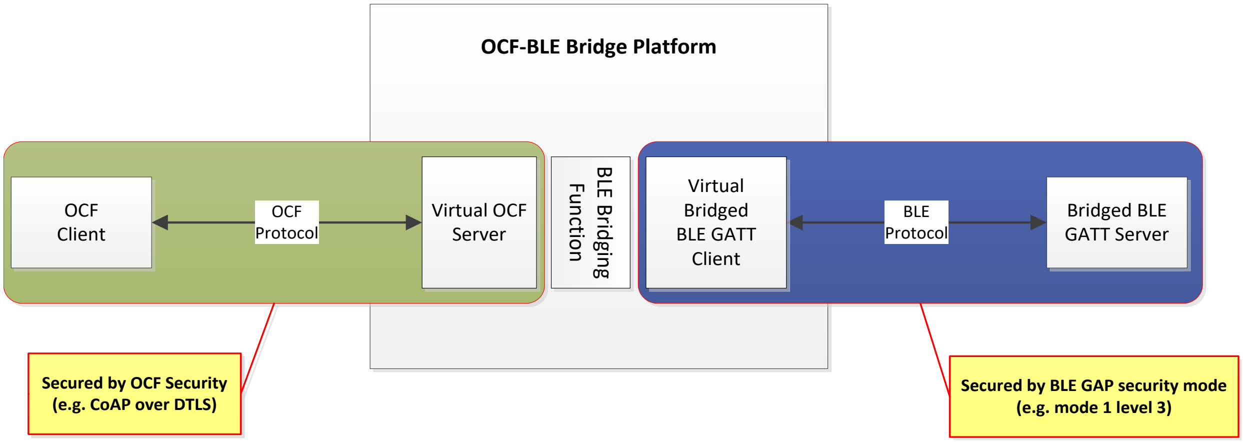

OCF-BLE Bridge Platform
OCF Client
OCF Protocol
Virtual OCF Server
BLE Bridging Function
Virtual Bridged BLE GATT Client
BLE Protocol
Bridged BLE GATT Server
Secured by OCF Security (e.g. CoAP over DTLS)
Secured by BLE GAP security mode (e.g. mode 1 level 3)

Figure E-1 Security Considerations for BLE Bridge

E.3 Security Considerations specific to the oneM2M Protocol

This clause intentionally left empty.

E.4 Security Considerations specific to the U+ Protocol

A U+ server supports one of the TLS 1.2 cipher suites as in Table E.2 defined in IETF RFC 5246.

Table E.2 TLS 1.2 Cipher Suites used by U+

Cipher Suite
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CCM
TLS_RSA_WITH_AES_256_CCM_8
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_CCM
TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CCM
TLS_DHE_RSA_WITH_AES_256_CCM_8

The security of the Haier U+ Protocol is proprietary, and further details are presently unavailable.

E.5 Security Considerations specific to the Z-Wave Protocol

Z-Wave currently supports two kinds of security class which are S0 Security Class and S2 Security Class, as shown in Table E.3. Bridged Z-wave Servers using S2 Security Class for communication with a Virtual Bridged Client would typically be considered secure from an OCF perspective. The appropriate selection for S2 Security Class and Class Name is left to the vendor.

Figure E-2 presents how OCF Client and Bridged Z-Wave Server communicate based upon their own security.

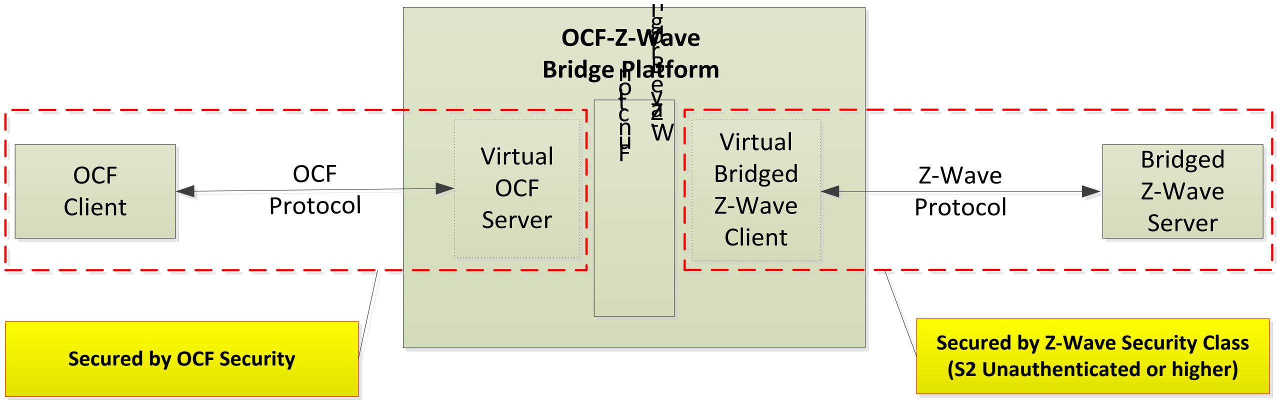


Figure E-2 Security Considerations for Z-Wave Bridge

All 3 types of S2 Security Class such as S2 Access Control, S2 Authenticated and S2 Unauthenticated provides the following advantages from the security perspective;

- The unique device specific key for every secure device enables validation of device identity and prevents man-in-the-middle compromises to security
- The Secure cryptographic key exchange methods during inclusion achieves high level of security between the Virtual Z-Wave Client and the Bridged Z-Wave Server.
- Out of band key exchange for product authentication which is combined with device specific key prevents eavesdropping and man-in-the-middle attack vectors.

See Table E.3 for a summary of Z-Wave Security Classes.

Table E.3 Z-Wave Security Class

Security Class	Class Name	Validation of device identity	Key Exchange	Message Encapsulation
S2	S2 Access Control	Device Specific key	Out-of-band inclusion	Encrypted command transmission
	S2 Authenticated	Device Specific key	Out-of-band inclusion	Encrypted command transmission
	S2 Unauthenticated	Device Specific key	Z-wave RF band used for inclusion	Encrypted command transmission
S0	S0 Authenticated	N/A	Z-wave RF band used for inclusion	Encrypted command transmission

On the other hand, S0 Security Class has the vulnerability of security during inclusion by exchanging of temporary ‘well-known key’ (e.g. 1234). As a result of that, it could lead the disclosure of the network key if the log of key exchange methods is captured, so Z-Wave devices might be no longer secure in that case.

E.6 Security Considerations specific to the Zigbee Protocol

The Zigbee 3.0 stack supports multiple security levels. A security level is supported by both the network (NWK) layer and application support (APS) layer. A security attribute in the Zigbee 3.0 stack, "nwkSecurityLevel", represents the security level of a device.

The security level nwkSecurityLevel > 0x04 provides message integrity code (MIC) and/or AES128-CCM encryption (ENC). Zigbee Servers using nwkSecurityLevel > 0x04 would typically be considered secure from an OCF perspective. The appropriate selection for nwkSecurityLevel is left to the vendor.

See Table E.4 for a summary of the Zigbee Security Levels.

Table E.4 Zigbee 3.0 Security Levels to the Network, and Application Support layers

Security Level Identifier	**Security Level** Sub-Field	Security Attributes	Data Encryption	**Frame Integrity** (Length of M of MIC, in Number of Octets)
0x00	'000'	None	OFF	NO (M=0)
0x01	'001'	MIC-32	OFF	YES(M=4)
0x02	'010'	MIC-64	OFF	YES(M=8)
0x03	'011'	MIC-128	OFF	YES(M=16)

0x04	'100'	ENC	ON	NO(M=0)
0x05	'101'	ENC-MIC-32	ON	YES(M=4)
0x06	'110'	ENC-MIC-64	ON	YES(M=8)
0x07	'111'	ENC-MIC-128	ON	YES(M=16)

Figure E-3 shows how communications in both ecosystems of OCF-Zigbee Bridge Platform are secured by their own security.

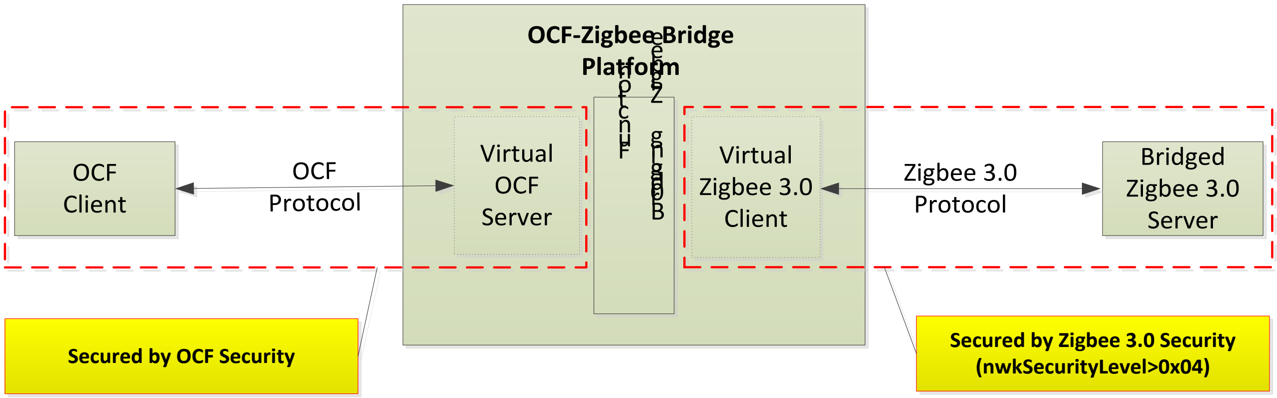

OCF-Zigbee Bridge Platform
OCF Client
OCF Protocol
Virtual OCF Server
Virtual Zigbee 3.0 Client
Zigbee 3.0 Protocol
Bridged Zigbee 3.0 Server
Secured by OCF Security
Secured by Zigbee 3.0 Security (nwkSecurityLevel>0x04)

Figure E-3 Security Considerations for Zigbee Bridge

E.7 Security Considerations specific to the the EnOcean Radio Protocol

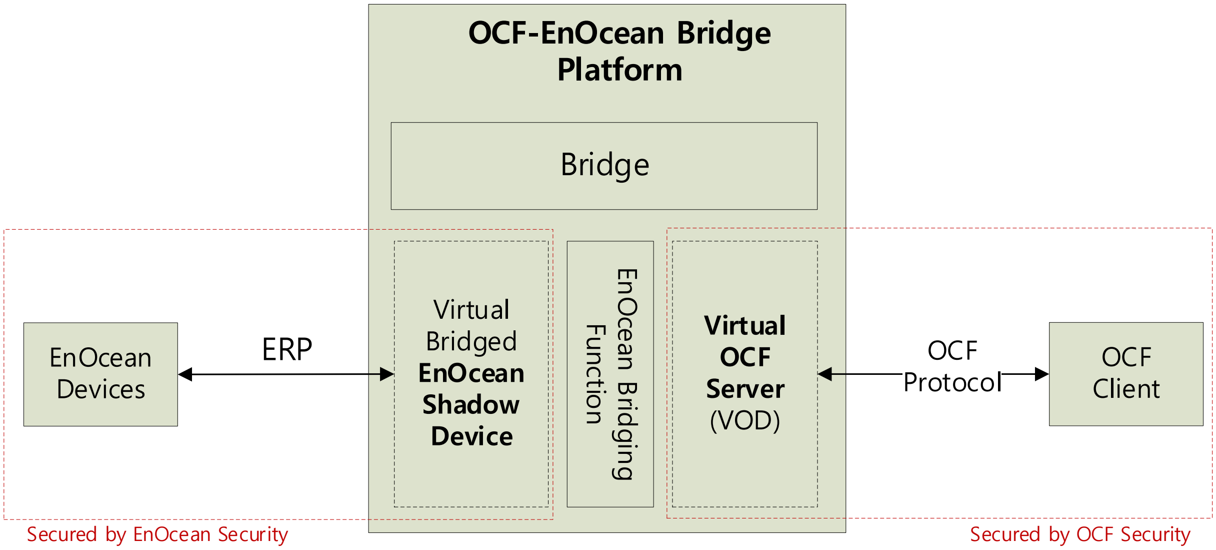
The EnOcean Radio Protocol supports four different security levels. The security level depends on which security mechanisms are used. Table E.5 defines them

Table E.5 EnOcean Radio Protocol security levels

Level	**Features**	**Replay Attack Vulnerability**	**Eavesdropping Vulnerability**
0	No Features (Unsecure)	Yes	Yes
1	With Encryption only	Yes	No
2	Without Encryption but with RLC and CMAC	No	Yes
3	With Encryption, RLC and CMAC	No	No

The security levels 1 and 2 have been declared deprecated and shall not longer be used. Security level 3 uses Variable AES Encryption, Rolling Code (RLC) and a cipher-based message authentication code (CMAC) with private keys and public vectors. Technically each feature can be combined with every other feature, even if it is obsolete or unreasonable.

Figure E-4 shows how communications in both ecosystems of OCF- EnOcean Bridge Platform are secured by their own security


OCF-EnOcean Bridge Platform
Bridge
EnOcean Devices
ERP
Virtual Bridged EnOcean Shadow Device
EnOcean Bridging Function
Virtual OCF Server (VOD)
OCF Protocol
OCF Client
Secured by EnOcean Security
Secured by OCF Security

8067

8068 **Figure E-4 Security Considerations for EnOcean Bridge**

8069