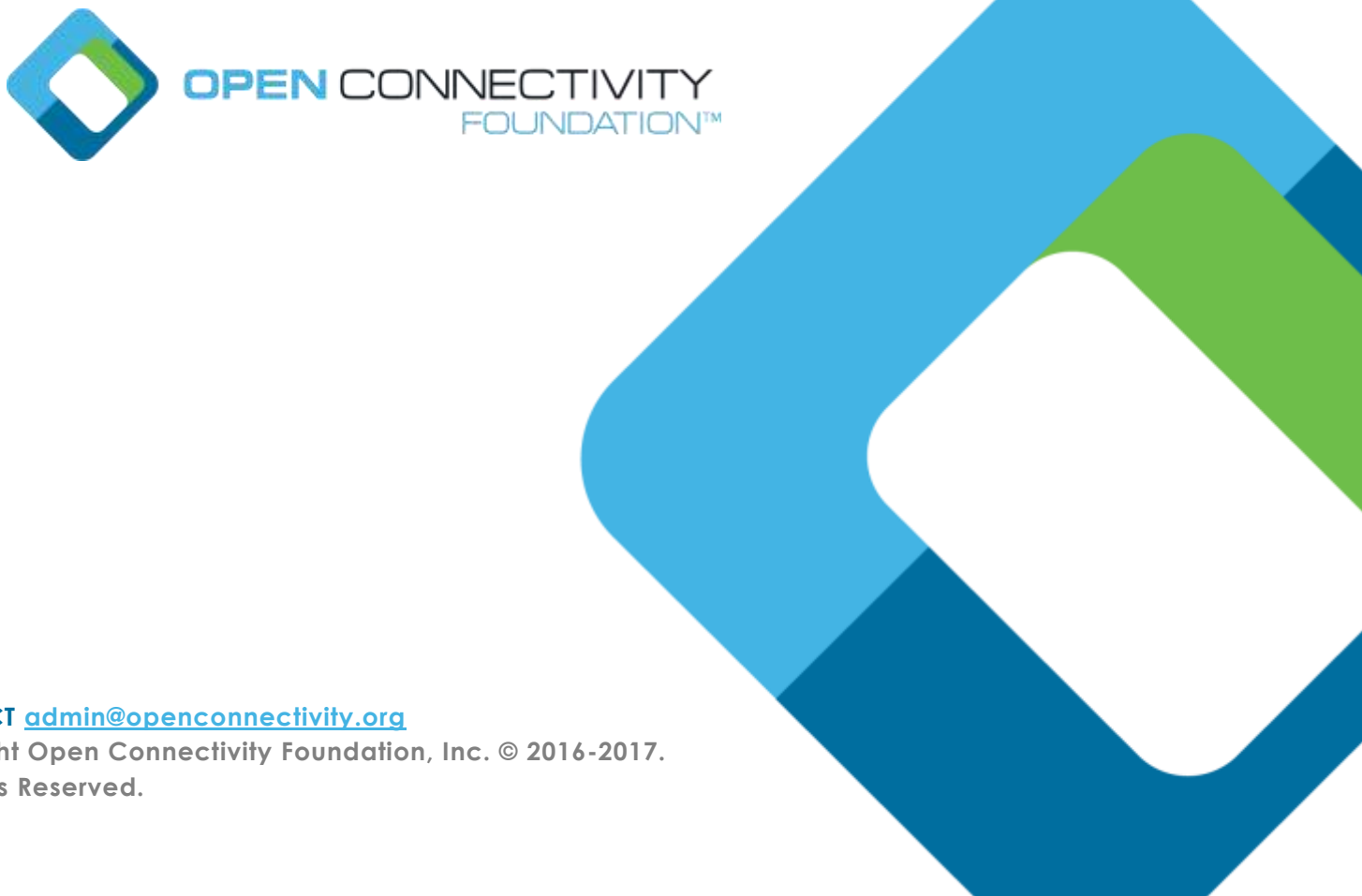
OCF Security Specification

VERSION 1.0.0_Korean | June 2017



CONTACT admin@openconnectivity.org

법적 고지 사항

본 문서에 기재된 내용 중 그 어느 것도 명시적 또는 암시적으로 기재 내용에 있어서 어떠한 형태의 사용 허가를 부여하거나 본 문서의 작성자 또는 개발자 중 어느 누구가 소유 또는 관할하는 어떠한 지식재산에 대해 어떠한 형태의 사용 허가도 부여하는 것을 의미하지 않습니다. 여기에 포함된 정보는 "있는 그대로" 제공되며, 적용 가능한 법에 의해 허용되는 최대 한도까지 본 스펙의 작성자 및 개발자는 특정한 목적을 위한 판매 적격성 또는 적합성의 암시적 보증을 포함하지만 이에 한정되지 않는 명시적 또는 암시적인 성문법 또는 불문법 상의 기타 모든 보증 및 조건에 대해 일절 책임을 지지 않습니다. OPEN CONNECTIVITY FOUNDATION, INC.는 비침해, 정확성, 또는 바이러스 비 감염에 대한 모든 보증에 대해서도 일절 책임을 지지 않습니다.

OCF 로고는 미국 및 다른 국가에서 Open Connectivity Foundation, Inc 의 상표입니다. *그 밖의 명칭 및 상표는 해당하는 소유자의 자산일 수 있습니다.

For Translation to Local Language

· 본 OCF 스펙 번역 버전은 OCF 기반의 제품 개발을 장려하고 이에 도움이 되도록 규범적인 영문 원본 버전으로부터 작성되었습니다. 영문 스펙의 정확한 번역을 위한 모든 노력을 기울이기는 하였지만 본 번역 버전을 규범적으로 간주해서는 안됩니다. OCF 인증 프로그램은 명백하게 영문 스펙을 기준으로 개발되어야 하며, 어떠한 면제 또는 면책 요구도 영문 스펙의 문구를 기준으로 평가되어야 합니다.

· 최신 영문 버전 스펙의 공개로부터 번역 버전의 공개까지는 소정의 지연이 있을 수 있습니다.

· OCF 스펙의 최신 영문 버전 및 해당 번역 버전에 관해서는
https://openconnectivity.org/developer/specifications 를 참조하여 주십시오.

목차

298 도면

341

342 표

1 적용 범위

본 스펙은 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 의 OCF 기본 계층에 영향을 주는 보안 목적, 철학, resource 및 메커니즘을 정의한다. 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 은 유익한 보안 컨텐츠를 포함한다. OCF Security 스펙은 보안 규범 컨텐츠를 포함하며 OCF 기본 또는 그 밖의 OCF 스펙에 관련된 유익한 컨텐츠를 포함할 수 있다.

2 인용 표준

다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification, version 1.1, Open Connectivity Foundation, October 11, 2016. Latest version available at: https://openconnectivity.org/specs/OIC_Core_Specification_v1.1.0.pdf.

OIC Smart Home Device Specification, version 1.1, Open Connectivity Foundation, October 11, 2016. Latest version available at: https://openconnectivity.org/specs/OIC_SmartHome_Device_Specification_v1.1.0.pdf.

OIC Resource Type Specification, version 1.1, Open Connectivity Foundation, October 11, 2016. Latest version available at: https://openconnectivity.org/specs/OIC_Resource_Type_Specification_v1.1.0.pdf.

JSON SCHEMA, draft version 4, JSON Schema defines the media type "application/schema+json", a JSON based format for defining the structure of JSON data. JSON Schema provides a contract for what JSON data is required for a given application and how to interact with it. JSON Schema is intended to define validation, documentation,

hyperlink navigation, and interaction control of JSON Available at: http://json-schema.org/latest/json-schema-core.html.

RAML, Restful API modelling language version 0.8. Available at: http://raml.org/spec.html.

3 용어, 정의, 기호 및 약어

본 스펙에서 사용되는 용어, 정의, 기호, 및 약어는 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 에 의해 정의된다. 규범적 보안 메커니즘 고유의 용어는 본 스펙에서 상황에 맞게 정의된다.

본 섹션에서는 이해를 돕기 위해 본 문서 또는 다른 OCF 스펙에서 정의된 용어를 재 기술한다. 이러한 용어의 기술은 규범이 아닌 단순 정보로 제공된다.

3.1 용어 및 정의

3.1.1

Access Manager Service

액세스 매니저 서비스는 Device Resource request 에 대한 응답으로 ACL Resource 를 동적으로 구성한다. 액세스 매니저 서비스는 원격으로 액세스 정책을 평가하고 미결 액세스 요청을 허가 또는 거절하는 Server 에 평가 결과를 제공한다.

3.1.2

ACL Provisioning Service

ACL Resource 의 프로비저닝 권한을 갖는 Device 에 부여된 명칭 및 Resource Type (oic.sec.aps).

3.1.3

Bootstrap Service

oic.sec.bss 타입의 서비스를 구현하는 Device.

3.1.4

Client

주 1: 상세 내용은 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 에 정의되어 있음.

3.1.5

Credential Management Service

크리덴셜 Resource 의 프로비저닝 권한을 갖는 Device 에 부여된 명칭 및 Resource Type (oic.sec.cms).

3.1.6

Device

주 1: 상세 내용은 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 에 정의되어 있음.

3.1.7

Device Class

RFC 7228 에 정의된 바와 같이, RFC 7228 은 OCF 소형 풋프린트 스택 사용 시에 대형 풋프린트 스택 을 구별하는 제한된 device 의 클래스를 정의한다. Class 2 이하가 소형 풋프린트 스택에 해당한다.

3.1.8

Device ID

스택 인스턴스 식별자.

3.1.9

Entity

주 1: 상세 내용은 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 에 정의되어 있음.

3.1.10

Interface

주 1: 상세 내용은 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 에 정의되어 있음.

3.1.11

Intermediary

Client 와 Server 의 role 을 둘 다 구현하는 Device 로 프로토콜 변환, 가상 device 의 물리적 device 에의 매핑, 또는 Resource 변환을 수행할 수 있다.

3.1.12

OCF Cipher Suite

Device 의 암호화 기능을 정의하는 알고리즘과 파라미터의 집합. OCF Cipher Suite 는 공개 키 그룹 동작, 서명, 및 공개 키를 지원하는데 사용되는 특정 해싱 및 인코딩을 포함한다.

3.1.13

Onboarding Tool

특정 device 에 대해 소유권을 설정하고 해당 device 를 네트워크 내에서 동작 가능 상태로 하는 특정 IoT 네트워크 내의 논리적 개체.

3.1.14

Out of Band Method

OCF 에 의해 지정되지 않은, 한쪽에서 다른 쪽으로 비밀을 전달하기 위한 모든 메커니즘.

3.1.15

Owner Credential

이어지는 상호 작용 동안에 Device 와 온보딩 툴 간의 상호 인증을 목적으로 온보딩 중에 온보딩 툴에 의해 Device 로 제공되는 크리덴셜.

3.1.16

Platform ID

주 1: 상세 내용은 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 에 정의되어 있음.

3.1.17

Property

주 1: 상세 내용은 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 에 정의되어 있음.

3.1.18

Resource

주 1: 상세 내용은 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 에 정의되어 있음.

3.1.19

Role (network context)

Device 의 전형적인 동작으로 Client, Server, 및 Intermediary 중 하나가 된다.

3.1.20

Role (Security context)

Device Resources 에의 액세스 시도 시에 Device 가 행사 가능한 role 에 명칭을 부여하는 OCF credentials Resource 의 Property. role 대 device UUID 를 통해 액세스를 시도하면 Client 에 따라 액세스 정책이 달라질 수 있다. 본 문서에서는 별도 기재가 없는 한 보안 컨텍스트를 가정한다.

3.1.21

Secure Resource Manager

ACL, 크리덴셜, 및 Device 소유권 이전 상태와 같은 security Resource 의 관리를 포함하는 보안 기능성을 구현하는 OCF Core 내의 모듈.

3.1.22

Security Virtual Resource

SVR 은 resource 지원 보안 기능이다.

564 **3.1.23**

565 **Server**

566 주 1: 상세 내용은 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다.
567 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정
568 내용 포함)이 적용된다.

569

570 OIC Core Specification 에 정의되어 있음.

571 **3.1.24**

572 **Trust Anchor**

573 두 암호화 개체 (예: Device 와 온보딩 툴)가 신뢰할 수 있도록 하는 신뢰 계층 내의 제대로 정의된
574 공유 권한.

575 **3.1.25**

576 **Unique Authenticable Identifier**

577 공개 키의 해시와 DeviceID 생성에 사용된 관련된 OCF Cipher Suite 로부터 생성되는 고유 식별자.
578 UAID 의 소유권은 피어 Device 에 의해 인증될 수 있다.

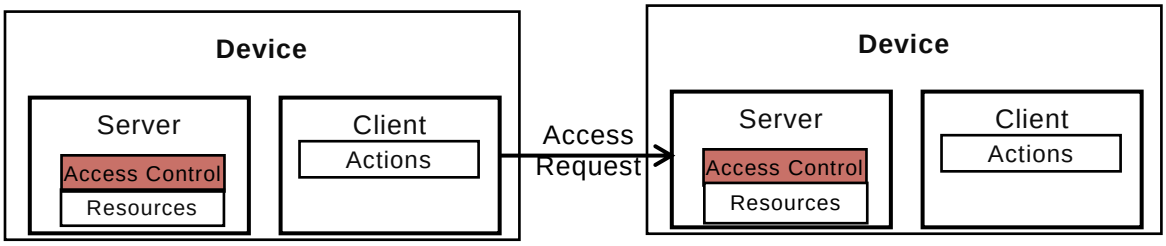
579 **3.2 기호 및 약어**

기호	설명
ACE	Access Control Entry
ACL	Access Control List
AMS	Access Manager Service
APS	ACL Provisioning Service
BSS	Bootstrap Service
CMS	Credential Management Service
CRUDN	CREATE, RETREIVE, UPDATE, DELETE, NOTIFY
CSR	Certificate Signing Request
ECDSA	Elliptic Curve Digital Signature Algorithm
EPC	Embedded Platform Credential
DPKP	Dynamic Public Key Pair
OC	Owner Credential
OCSP	Online Certificate Status Protocol
OBT	Onboarding Tool
PIN	Personal Identification Number

PSI	Persistent Storage Interface
RNG	Random Number Generator
SACL	Signed Access Control List
SE	Secure Element
SRM	Secure Resource Manager
SVR	Security Virtual Resource
TEE	Trusted Execution Environment
UAID	Unique Authenticable Identifier

580 **표 1 – 기호 및 약어**

581 ## 3.3 협약



582 **도 1 – OCF 상호 작용**

583 Device 는 Server 에 대해 동작을 수행하는 Client role 을 구현할 수 있다. 동작은 Server 에 의해
584 관리되는 Resource 에 액세스한다. OCF 스택은 Resource 에 대한 액세스 정책을 행사한다. 단대단
585 Device 상호 작용은 세션 보호 프로토콜 (예: DTLS) 또는 데이터 암호화 방법을 사용해서 보호할 수
586 있다.

587

4 문서 규약 및 구성

본 문서는 OCF core framework 및 애플리케이션을 위한 보안 구현에 사용되는 Resource, 프로토콜, 및 규약을 기술한다.

본 문서를 위해 **Error! Reference source not found.** 내의 용어 및 정의가 적용된다.

4.1 표기법

본 스펙에서 기능은 다음과 같이 필수(Required), 권고(Recommended), 허가(Allowed), 또는 사용 금지(DEPRECATED)로 분류된다.

필수 (강제 또는 의무적)

이러한 기본 기능은 OCF Core Architecture 를 준수하도록 구현되어야 한다. "하지 말아야 한다"나 "금지한다" 등의 구절은 금지되는, 즉, 수행하는 경우 구현이 스펙을 준수하지 않음을 의미하는 행위를 나타낸다.

권고 (또는 제안)

이러한 기능은 OCF Core Architecture 에 의해 지원되는 기능을 부가하며 구현되어야 한다. 권고 기능은, 통상적으로 중대한 복잡성의 증가 없이 OCF Core Architecture 의 기능을 이용한다. 규정 준수 테스트를 위해 권고 기능이 구현된다면 이 가이드라인에 따르는 특정 요건을 만족해야 한다. 일부 권고 기능은 추후에 필수 요건이 될 수 있다. "하지 않는 것이 좋다"라는 표현은 허용되지만 권고하지 않는 작용을 나타낸다.

허가 (또는 허용)

이러한 기능은 OCF Core Architecture 에 의해 필수적이지도 않을 뿐더러 권고되지도 않지만 기능이 구현된다면 이 가이드라인에 따르는 특정 요건을 만족해야 한다.

조건부 허용 (CA)

정의 또는 행위는 조건에 의존한다. 특정 조건이 만족되면 정의 또는 행위가 허용되고 그렇지 않으면 허용되지 않는다.

조건부 필수 (CR)

정의 또는 행위는 조건에 의존한다. 특정 조건이 만족되면 정의 또는 행위가 필수로 된다. 그렇지 않으면 특별한 기재가 없는 한 default 로 허용된다.

사용 금지

이에 해당하는 기능은 본 스펙에서 설명은 하고 있지만 역 호환성을 제외하고는 구현되어서는 안된다. 현재 스펙에 따르는 동작 동안 사용 금지된 기능의 발생은 구현 동작에 어떤 영향도 끼치지 않으며 어떠한 에러 상태도 생성하지 않는다. 역 호환성은 기능이 구현되고 특정된 대로 기능할 것을 요구할 수 있지만 본 스펙에 따르는 구현에 의해 사용되어서는 안된다.

문자 그대로 해석되는 스트링은 "인용부호"를 사용한다.

강조하는 단어는 *이탤릭체*로 표기한다.

4.2 Data type

- 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 참조.

4.3 문서 구조

참고 섹션은 개요를 참조하고 규정 섹션은 그 밖의 섹션을 참조하기 바란다.

Security 스펙은 RAML 을 스펙 언어로 사용하고 JSON Schema 를 모든 CRUDN 동작에 대한 페이로드 정의로 사용한다. CRUDN 동작의 매핑은 다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

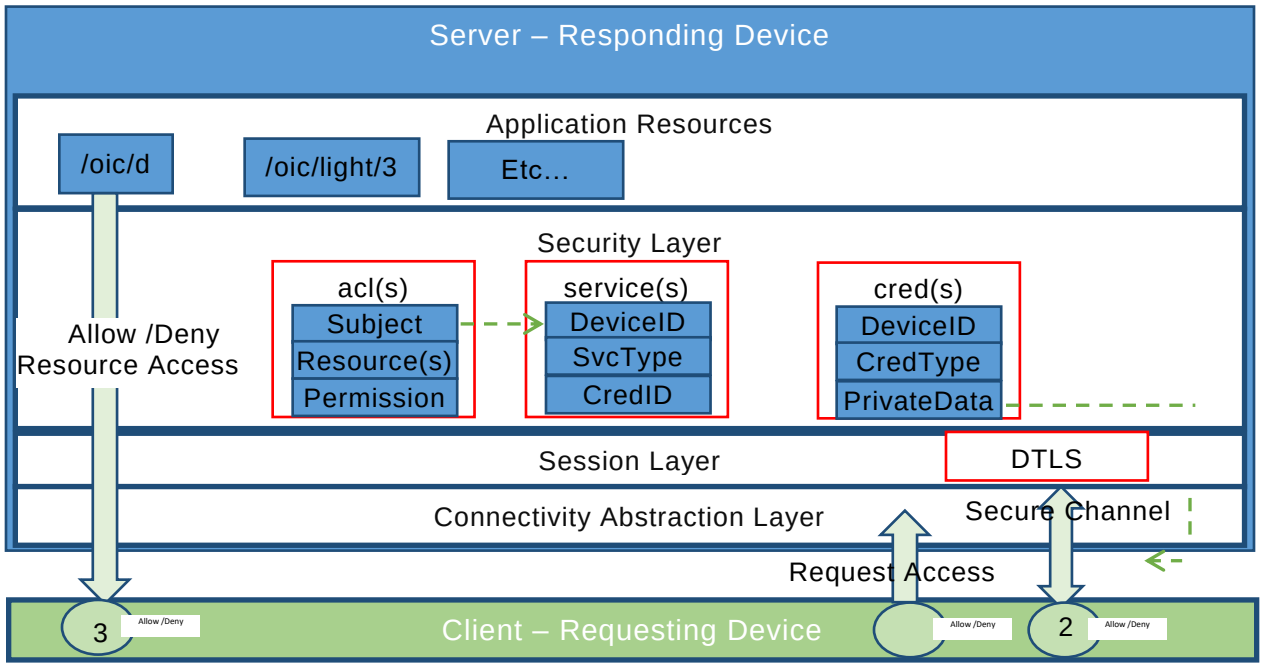
OIC Core Specification 에 규정된다.

645

5 보안 개요

이 섹션은 참조 섹션이다. OCF 보안 아키텍처의 목적은 Resource 및 Resource 보호를 지원하기 위해 사용되는 HW 와 SW 의 모든 측면을 보호하기 위한 것이다. OCF 의 관점에서 Device 는 OCF 스펙을 준수하는 논리적 개체이다. Device 간의 상호 작용에 있어서, Server 로 동작하는 Device 는 Resource 를 보유 및 제어하며 일련의 보안 메커니즘에 입각해서 Client 로 동작하는 Device 에 이러한 Resource 에 대한 액세스를 제공한다. Device 를 받아들이는 Platform 은 본 스펙에서 설명되는 다양한 동작의 견고성을 확보하기 위해 요구되는 보안 강화를 제공할 수 있다.

동작의 보안 이론은 다음과 같은 단계로 설명할 수 있다.



도 2 – OCF 계층

1. Client 가 Server (Resource 를 보유하는 Device)와의 네트워크 연결을 확립한다. 연결성 추상화 계층은 연결성 옵션의 차이에도 불구하고 Device 가 확실하게 연결 가능하도록 한다.

2. Device (예: Server 및 Client)가 서로 간의 상호 인증 보안 채널을 통하거나 또는 통하지 않고 메시지를 교환한다.

   - 각 Device 상의 oic.sec.cred Resource 는 상호 인증 및 인증서 검증에 (사용 가능 시에) 사용되는 크리덴셜을 보유한다.

- 보안 채널을 통해 수신한 메시지는 deviceUUID 와 관련 지어진다. 인증서 크리덴셜의 경우, deviceUUID 는 다른 Device 로부터 수신한 인증서 내에 존재한다. 대칭 키 크리덴셜의 경우, deviceUUID 는 oic.sec.cred Resource 내의 크리덴셜로 구성된다. device 컨텍스트와 Device 를 구현하는 Platform 간에는 바인딩이 존재한다.

- Server 는 Client 와 임의의 수의 허가 roleid 를 관련 지을 수 있다. 인증서를 사용한 상호 인증의 경우, 허가 roleid (존재하는 경우)는 role 인증서 내에 제공되며, 이는 Server 에 대해 Client 에서 구성된다. 대칭 키의 경우, 허가 roleid (존재하는 경우)는 oic.sec.cred 내의 크리덴셜로 구성된다.
- 비 보안 채널을 통해 Server 가 수신한 Request 는 익명으로 취급되어 deviceUUID 또는 roleid 와 관련 지어지지 않는다.

3. Client 가 Server 로 request 를 제출한다.

   - request 가 보안 채널을 통해 전송되면 Client 는 request 에 'role' 옵션을 포함함으로써 특정 roleid 를 명시적으로 주장하거나 'role' 옵션을 포함하지 않음으로써 Client 에 관련 지어진 모든 roleid 를 암시적으로 주장할 수 있다.

4. Server 가 request 를 수신한다.

   a. request 가 비 보안 채널을 통해 전송되면 Server 는 request 를 익명으로 취급하여 deviceUUID 나 roleid 가 request 에 관련 지어지지 않는다.
   b. Request 가 보안 채널을 통해 수신되면 Server 는 deviceUUID 를 관련 짓고, Client 의 허가 roleid 에 매칭시킴으로써 명시적으로 주장된 roleid 를 유효하게 하거나 Client 의 모든 유효 roleid 를 암시적으로 주장한다.
   c. 그리고 나서, Server 는 Access Control List (ACL)에서 다음의 기준에 부합하는 ACL 엔트리를 찾는다.
      - 요청된 Resource 가 ACE 내의 Resource reference 에 매칭된다,
      - 요구된 동작이 ACE 의 "승인"에 의해 허가된다, 및
      - "subjectUUID"가 모든 Device 에 매칭되는 특수 와일드카드 값을 포함하거나, Device 가 익명이 아닌 경우, 대상이 Client Deviceid 또는 주장된 유효 roleID 에 매칭된다. 특별 액세스가 필요한 경우, 요청자는 role 을 주장할 수 있다.

      매칭되는 ACE 가 있으면 Resource 에의 액세스가 허가되고, 그렇지 않으면 액세스가 거부된다. 액세스는 Server 의 Secure Resource manager (SRM)에 의해 실시된다.

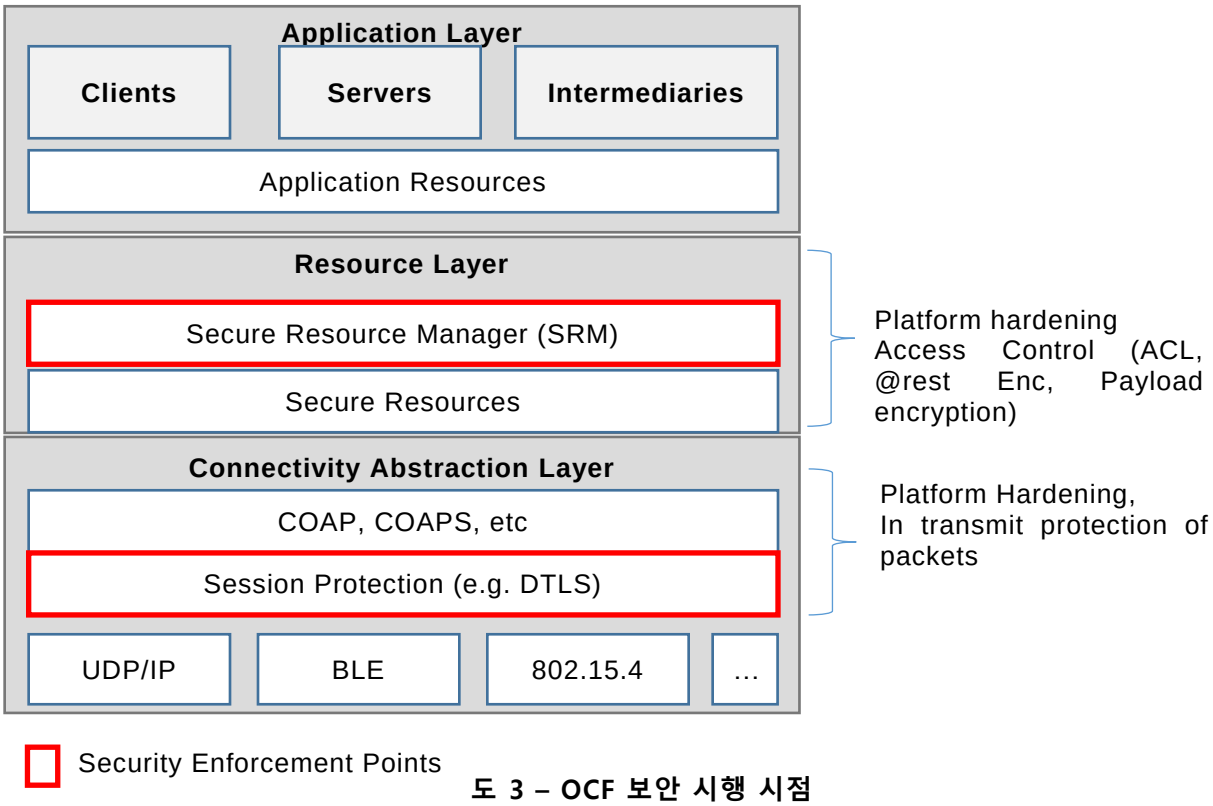
Resource 보호는 유휴 및 전송 중의 데이터의 보호를 포함한다. 액세스 제어 메커니즘과는 별도로, OCF Security 스펙은 Server 에 저장되는 동안 Resource 의 보안 저장에 관한 사양을 포함하지 않음에 주의해야 한다. 단, Secure storage 와 액세스 제어의 조합을 통해 보안 Resource 의 유휴

보호가 제공될 예정이다. Secure storage 는 하드웨어 보안 또는 유휴 데이터의 암호화를 통해 구현할 수 있다. Secure storage 의 정확한 구현은 섹션 15 에 규정된 일련의 강화 요구 사항을 전제로 하며, 인증 가이드라인을 전제로 할 수 있다.

반면에, 전송 중인 데이터의 보호는 본 스펙의 규정 부분으로 완전하게 규정된다. 전송 중의 데이터는 다음과 같은 단계에서 보호할 수 있다.

1. 근본적인 전송 보안과는 관계 없이 payload 보호를 허용하는 JSON Web Encryption (JWE) 및 JSON Web Signature (JWS)와 같은 메커니즘을 통한 Resource 계층. 이것은 payload 보호를 위한 DTLS 를 이용할 수 없는 전송 메커니즘에 대해 필요한 경우가 있다.

2. DTLS 와 같은 메커니즘의 사용을 통한 전송 계층. DTLS 는 payload 전체에 대한 보호보다는 패킷 단위의 보호를 제공한다. 예를 들어, payload 전체로서의 무결성이 필요한 경우, 전송 계층으로 패킷을 보내기 전에 별도의 서명 메커니즘이 준비되어 있어야 한다.



도 3 – OCF 보안 시행 시점

5.1 액세스 제어

OCF framework 는 Server 가 Resource 를 가지고 있고 액세스 제어와 인증 메커니즘에 따라 Client 에게 사용 가능하도록 제공하는 것을 전제로 하고 있다. end point 에서의 Resource 는 액세스 제어, 인증, 및 기밀성 보호의 구현을 통해 보호된다. 이 섹션에서는 ACL 의 사용을 통한 Access Control (AC)의 개요를 설명한다. 단, OCF stack 에서의 AC 는 전송 및 연결성 추상화 계층과는 상관 없이 이루어진다고 볼 수 있다.

액세스 제어의 구현은 Resource 에 대한 액세스 정책 집합의 선험적인 정의에 의존한다. 정책은 Access Control Entry (ACE)의 형태로 로컬 ACL 또는 Access Manager Service (AMS)에 의해 저장될 수 있다. 각 ACE 는 부여된 허가의 선택적 유효 기간과 함께 특정 Resource 에 액세스하기 위한 허가를 정의한다. 다음과 같이 두 가지의 액세스 제어 메커니즘을 적용할 수 있다.

- Subject 기반 access control (SBAC). 각 ACE 는 Resource 에 대해 정의된 정책에 포함되는 subject 에 대해 요청하는 개체의 subject (예: 요청자의 아이덴티티)를 매칭한다. 요청자의 아이덴티티 확인에는 인증 프로세스를 필요로 한다.
- Role 기반 Access Control (RBAC). 각 ACE 는 액세스를 요구하는 개체에 의해 취해지는 role 에 대해 Resource 에 대한 정책에 의해 요구되는 role 을 매칭한다. 요청자의 role 확인에는 적절한 허가를 필요로 한다.

OCF 액세스 제어 모델에서 각 Resource instance 는 관련된 액세스 제어 정책을 갖고 있어야 한다. 이것은 Server 로 동작하는 각 Device 가 보호하는 Resource 에 대한 ACL 을 갖고 있어야 함을 의미한다. 매칭되는 ACE 가 없으면 Resource 에 액세스할 수 없게 된다.

ACE 는 적용할 ACE 에 대해 요구되는 subject (즉, OCF Client)와 Resource 모두에 매칭되어야 한다. 다음과 같이 다양한 방법을 통해 subject 를 매칭시킬 수 있다: (1) device id, (2) role, 또는 (3) wildcard. client 가 server 에 연결하는 방식은 액세스 제어 결정을 위한 관련된 컨텍스트일 수 있다. 인증된 대 비 인증된 및 암호화된 대 비 암호화된 연결에 대한 와일드카드 매칭은 액세스 정책이 subject class 에 광범위하게 적용될 수 있도록 한다.

와일드카드 매칭 정책 예:

```
"aclist2": [
  {
        "subject": {"conntype" : "anon-clear" },
```



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

ACL 의 형식에 대한 자세한 사항은 섹션 **Error! Reference source not found.**에 정의된다. ACL 은 하나 이상의 ACE 로 구성된다. ACL 은 Device 에 대한 액세스 제어 정책을 정의한다.

ACL Resource 는 SRM 와 PSI 에 의한 저장 및 처리 시에 다른 민감한 Resource 와 동일한 보안 보호를 필요로 하는 점에 주의해야 한다. 그러므로, ACL 의 보호에는 근본적인 Platform (HW 및 SW)의 강화가 고려되어야 하며, 아래에 설명된 바와 같이 ACL 은 상이한 적용 범위 레벨을 가질 수 있으므로, 각 적용 범위 레벨에 대해 강화가 이루어져야 한다. 예를 들어, 물리적 device 는 복수의 Device 구현을 받아들일 수 있으므로 동일한 Device 내 상이한 Server 를 위한 ACL 의 보안 저장, 사용, 및 분리가 고려되어야 한다.

5.1.1 ACL 아키텍처

Server 는 client 의 request 를 처리하기 전에 요청된 Resource 를 검사한다. 요청자 및 요청된 Resource 에 매칭되는 하나 이상의 ACE entry 를 찾기 위해 access control resource (예: /oic/sec/acl, /oic/sec/acl2 등)를 검색한다. 매칭되는 것을 찾으면 허가 및 기간 제약이 적용된다. 복수의 매칭되는 것을 찾으면 겹치는 기간에 대해 허가의 논리합이 적용된다.

server 는 연결 컨텍스트를 사용해서 subject 가 인증되었는지 및 데이터 기밀성이 적용되었는지 여부를 결정한다. 각각의 측면에 대해 subject 매칭 와일드카드 정책을 적용할 수 있다. 사용자가 인증되면 role 또는 device 아이덴티티를 토대로 세분화된 subject matching 이 이루어질 수 있다.

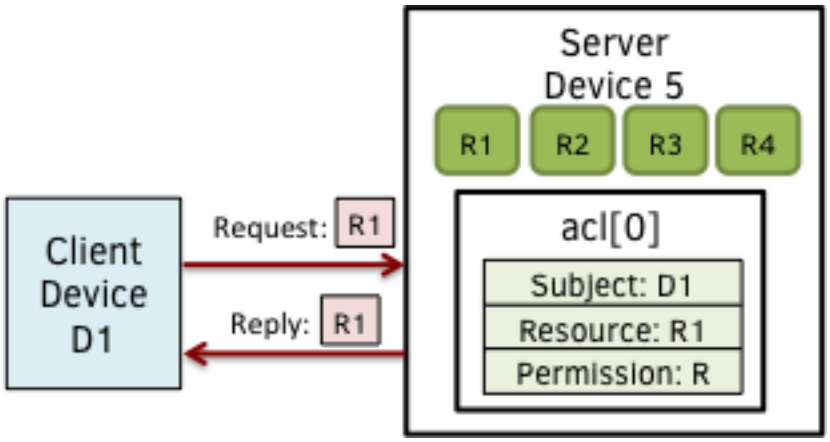
각 ACE 는 주어진 Resource 요청자에 대해 적용할 허가 집합을 포함한다. 허가는 CREATE, RETREIVE, UPDATE, DELETE, 및 NOTIFY (CRUDN) 동작의 조합으로 구성된다. 요청자는 Device 로서 인증되고 하나 이상의 role 로 선택적으로 동작한다. Device 는 role 를 행사할 때 격상된 액세스 허가를 취득할 수 있다. 예를 들어, ADMINISTRATOR role 은 일반적으로 액세스할 수 없는 부가적인 Resource 및 Interface 가 보이도록 한다.

5.1.1.1 로컬 ACL 의 사용

Server 는 기기 내 ACL Resource 를 가질 수 있다. 로컬 ACL 은 아래에 설명된 바와 같이 Access Management Service (AMS)에 의한 원격 ACL 처리에 비해 액세스 제어 처리에 있어서 더 큰 자율성을 가질 수 있다.

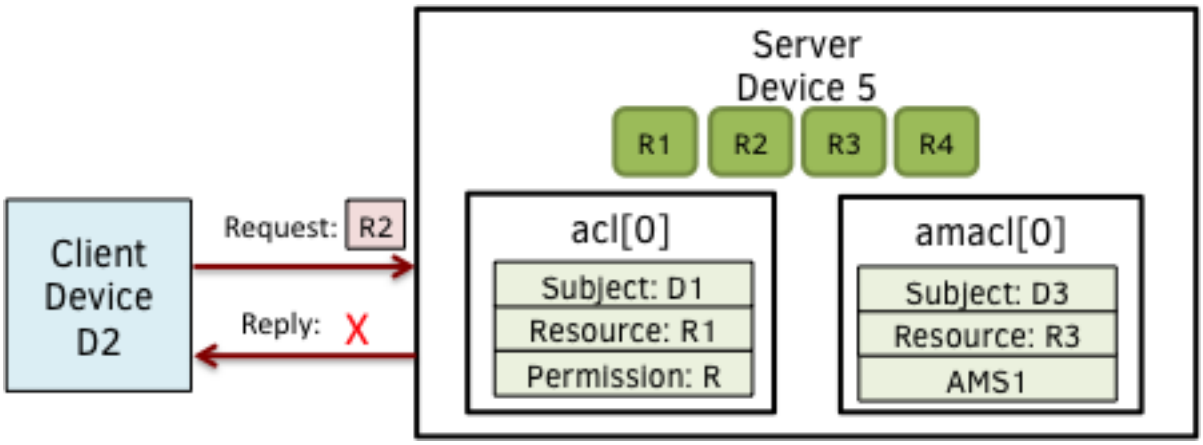
다음의 유스케이스는 액세스 제어 동작을 설명한다.

유스케이스 1: Server Device 는 4 개의 Resource (R1, R2, R3, 및 R4)를 호스트한다. Client Device D1 은 Server Device 5 에 호스트된 Resource R1 에 대한 액세스를 요청한다. ACL[0]은 아래의 Resource R1 에 해당하고 인가된 subject 로 D1 을 포함한다. 따라서, 로컬 ACL /oic/sec/acl/0 이 request 에 매칭되므로 Device D1 의 Resource R1 에 대한 액세스가 승인된다.



도 4 – 유스케이스-1: 단순 ACL 적용

유스케이스 2: Resource R2 에 관련된 subject D2 에 매칭되는 로컬 ACL 이 없고 AMS 정책을 찾을 수 없으므로 Client Device D2 의 액세스가 거부된다.



도 5 – 유스케이스 2: 요구되는 Resource 에 대한 정책이 없을 때

5.1.1.2 Access Manager Service 의 사용

AMS 는 ACL 정책 관리를 향상시킨다. 단, 이것은 중앙 실패점이 될 수 있다. 네트워크 지연 오버헤드로 인해 AMS 를 통한 ACL 처리 속도가 더 느려질 수 있다.

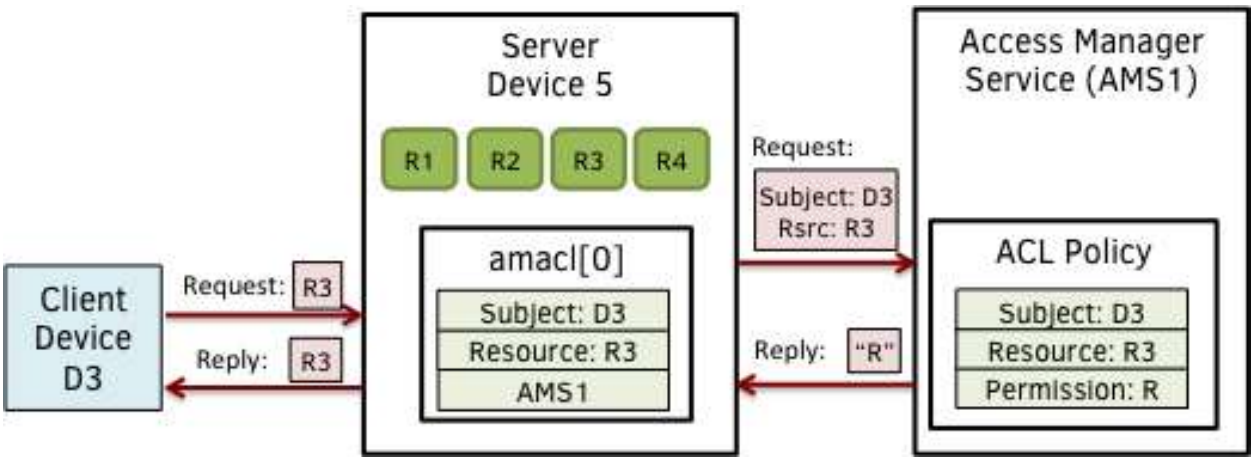
AMS 는 액세스 제어 결정을 집중 처리하지만 시행 의무는 Server Device 에 있다. Server 는 Resource 집합에 대해 사용할 ACL 메커니즘을 결정한다. /oic/sec/amacl Resource 는 액세스 결정을 위해 어떤 Resource 가 AMS 를 사용할지를 지정하는 ACL 구조이다. /oic/sec/amacl 은 local ACL (/oic/sec/acl)과 함께 사용될 수 있다.

AMS 는 /oic/sec/acl2.rowneruuid 에 포함된 device 식별자에 대해 발행된 크리덴셜을 참조해서 인증된다.

Server Device 는 /oic/sec/acl2.rowneruuid 에서 찾은 Device ID 를 사용해서 사전에 AMS 에 연결할 수 있다. 그렇지 않으면, Server 는 요청자로 하여금 SACL Resource /oic/sec/sacl 을 사용해서 적합한 ACE 정책을 취득하도록 지시하는 ACCESS_DENIED_REQUIRES_SACL 에러와 함께 Resource 액세스 요청을 거절할 수 있다. /oic/sec/sacl 서명은 /oic/sec/acl2.rowneruuid 와 관련된 크리덴셜 Resource 를 사용해서 인증될 수 있다.

다음의 유스케이스는 AMS 를 사용한 액세스 제어를 설명한다.

유스케이스 3: Device D3 는 Resource R3 에 대한 액세스를 요청하고, /oic/sec/amacl/0 이 Access Manager Server AMS1 서비스를 참조하기 위한 정책에 매칭되므로 허가 Perm1 으로 액세스 권한을 받는다.

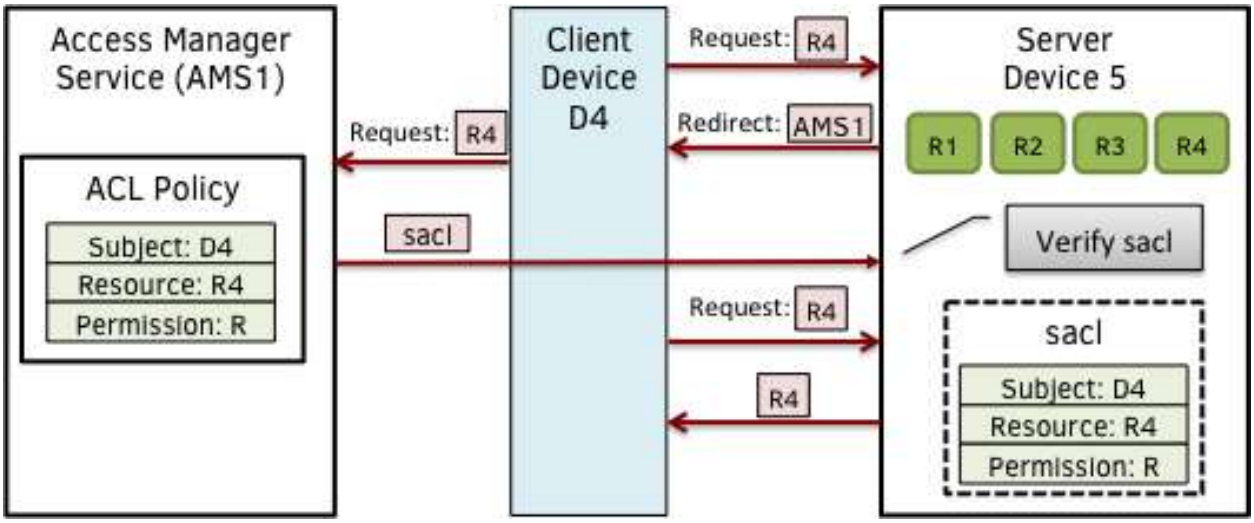


도 6 – 유스케이스-3 액세스 매니저 서비스 지원 ACL

유스케이스 4: Client Device D4 가 Server Device 5 의 Resource R4 에 대한 액세스를 요청하지만, 매칭되는 ACE 가 없으므로 AMS1 을 /oic/sec/sacl Resource 발행자로 식별하는 에러를 리턴함으로써 Client Device D4 를 AMS1 으로 리디렉션한다. Device D4 는 AMS1 에 의해 서명된 Sacl1 을 취득하여 SACL 을 Server D5 로 전송한다. D5 는 /oic/sec/sacl Resource 에서 서명을 확인하고 Perm2 액세스를 승인할 ACE 정책을 평가한다.

ACE 리디렉션은 D4가 매칭이 존재하지 않는다는 내용의 이유 코드 (즉, ACCESS_DENIED_NO_ACE)를 수신하면 발생할 수 있다. D4는 /oic/sec/acl2 Resource를 읽어서 AMS를 식별하기 위한 rowneruuid를 찾은 다음에 프로비저닝 request를 제출한다. 이 예에서, AMS는 SACL Resource의 공급을 선택하지만, 로컬 ACL Resource /oic/sec/acl 및 /oic/sec/acl2의 재 프로비저닝을 선택할 수도 있다. 결과적으로 request가 재 발행된다. D4는 Device 온보딩의 일부로서 또는 이어지는 크리덴셜 프로비저닝 동작을 통해 AMS에 도입된 것으로 간주된다.

그렇지 않으면 필요한 크리덴셜의 프로비저닝을 위해 Credential Management Service (CMS)가 참조될 수 있다.



834 **도 7 – 유스케이스-4 AMS 로부터 ACL 의 동적 취득**

835 ### 5.1.2 액세스 제어 범위 레벨

836 **그룹 레벨 액세스** – 그룹 범위는 특정 컨텍스트를 위해 그룹 지어진 Device 의 그룹에의 AC 의
837 적용을 의미한다. 그룹 크리덴셜은 그룹에의 데이터 암호화 또는 그룹에의 개별 Device 멤버 인증
838 시에 사용될 수 있다. 그룹 레벨 액세스는 모든 그룹 멤버가 그룹 데이터에 액세스할 수 있지만 그룹
839 멤버가 아닌 경우에는 명백한 액세스 권한을 부여 받아야 함을 의미한다. 그룹 레벨 액세스는
840 와일드카드 매칭을 사용해서 지정될 수도 있다.

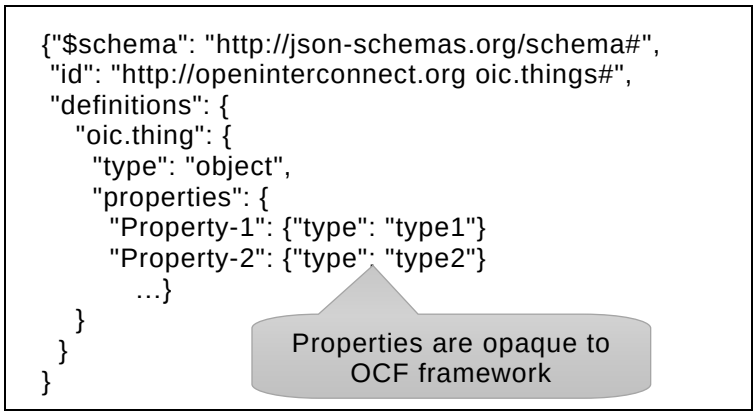
841 **OCF Device 레벨 액세스** – OCF Device 범위는 개별 Device 에의 AC 의 적용을 의미하며, 복수의
842 Resource 를 포함할 수 있다. Device 레벨 액세스는 DeviceID 에 의해 식별되는 Device 에서 사용
843 가능한 모든 Resource 에의 액세스 권한 확장을 암시한다. Device 에서 AC 메커니즘을 위해
844 사용되는 크리덴셜은 OCF Device 에 국한된다.

845 **OCF Resource 레벨 액세스** – OCF Resource 레벨 범위는 개별 Resource 에의 AC 의 적용을
846 의미한다. Resource 액세스에는 Resource 를 보유하는 개체 (Server)가 어떻게 액세스를 요청하는
847 개체 (Client)에게 Resource 에 액세스하도록 결정할지를 지정하기 위한 ACL 이 필요하다.

848 **Property 레벨 액세스** - Property 레벨 범위는 부모 Resource 의 일부인 property 에만 AC 를
849 적용함을 의미한다. 이것은 상이한 property 에 대해 상이한 허가를 필요로 하는 Resource 에 대해
850 보다 세분화된 AC 를 제공하기 위한 것이다. Property 레벨 액세스 제어는 단일 property 를
851 포함하는 Resource 를 생성함으로써 얻어진다. 이 기법을 사용하면 Resource 레벨 액세스 제어
852 메커니즘을 사용해서 보다 세분화된 레벨의 액세스를 시행할 수 있다.

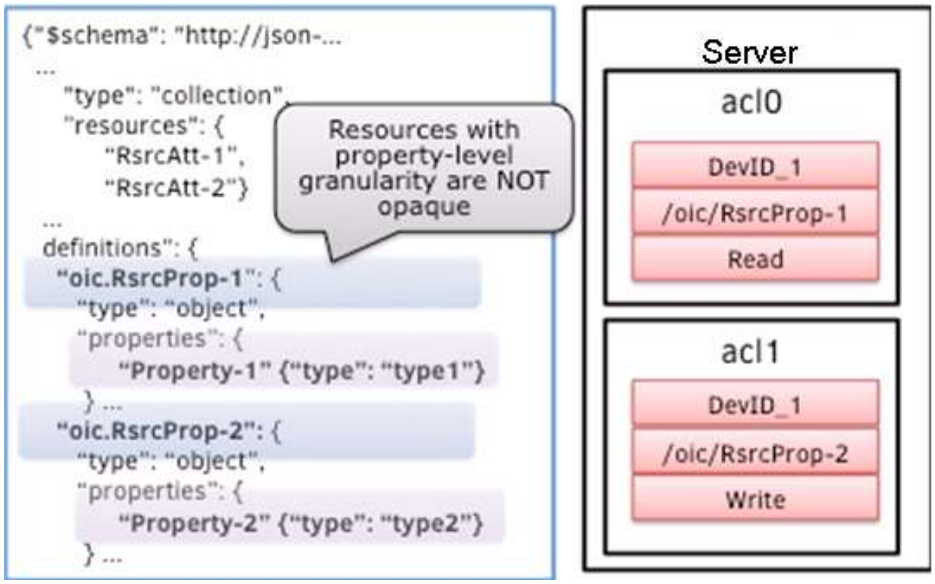
853

854 Resource 의 재 디자인이 현실적으로 불가능한 정적 Resource 에의 액세스 제어에는, 별도의
855 액세스 허가를 갖는 차일드 Resource 를 참조하는 collection Resource 를 도입하는 것이 적절할 수
856 있다. 아래에 보이는 예에서, "oic.thing" Resource 는 상이한 허가를 필요로 하는 Property-1 과
857 Property-2 의 두 개의 property 를 갖는다.



도 8 – 불투명한 Property 를 사용한 Resource 정의 예

현재, OCF framework 는 레벨 정보를 불투명한 것으로 적절하게 취급한다. 따라서, 상이한 허가를 ACL 정책의 일부로 할당할 수 없다 (예: Property-1 에의 읽기 전용 허가와 Property-2 에의 쓰기 전용 허가). 그러므로, "oic.thing"은 두 개의 신규 Resource "oic.RsrcProp-1"과 "oic.RsrcProp-2"로 분리된다. 이러한 식으로, property 레벨 ACL 은 Resource-level ACL 의 사용을 통해 얻을 수 있다.



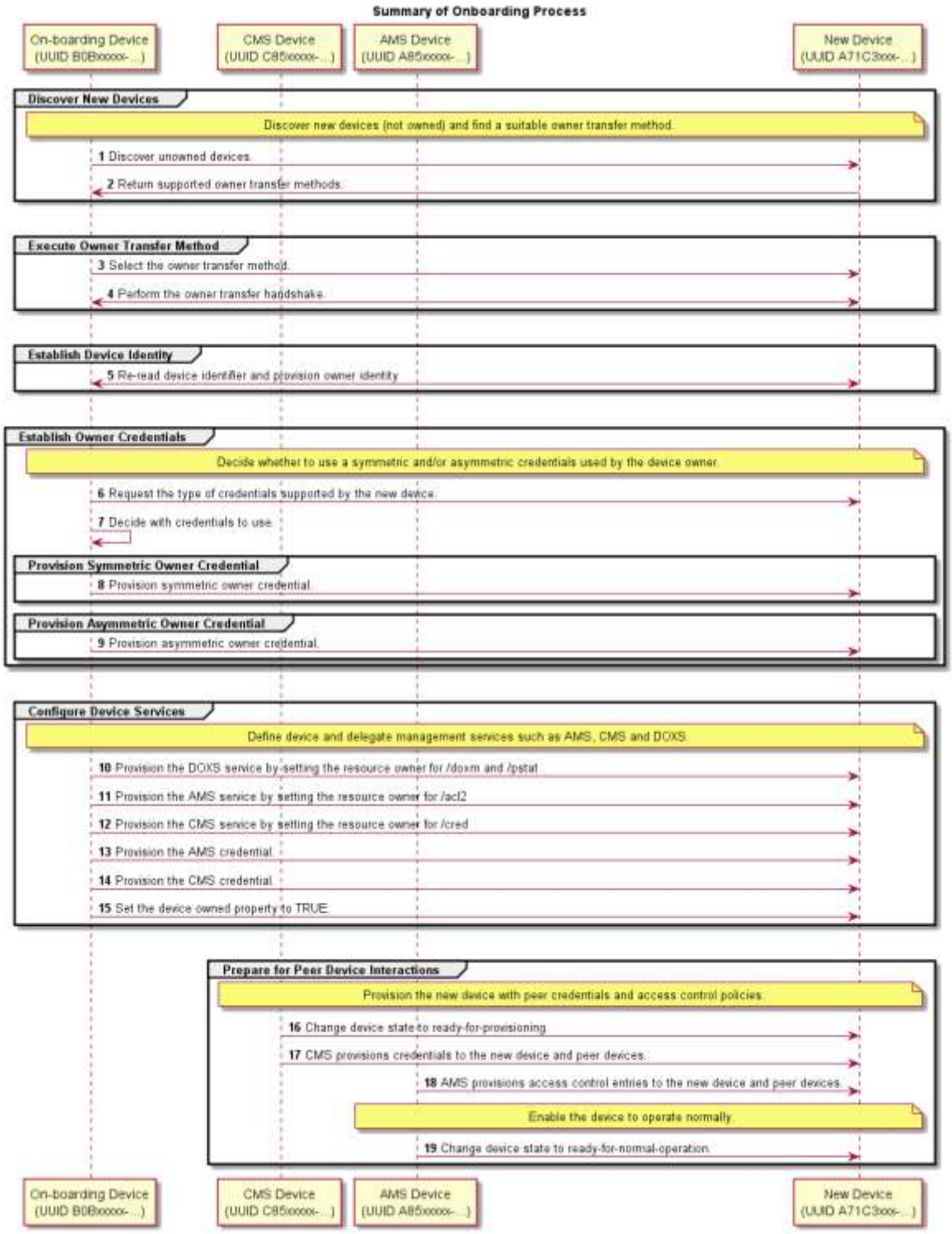
도 9 – Property 레벨 액세스 제어

5.2 온보딩 개요

Device 가 OCF 환경에서 작동 가능하고 다른 Device 와 연동될 수 있도록 하려면, 사전에 적절하게 온보딩 되어야 한다. Device 온보딩의 첫 번째 단계는 소유권을 설정하는 단계이다. Device 를 소유/구입한 정당한 사용자는 온보딩 툴 (OBT)을 사용하고, OBT 를 사용해서 하나의 소유권 이전 방법 (Owner Transfer Method: OTM)을 사용해서 소유권을 설정한다. 소유권이 설정되면, OBT 를

871 통해 Device 를 프로비저닝 할 수 있게 되며, 결과적으로 Device 가 OCF 환경에서 작동 가능하게
872 되고 다른 Device 와 연동할 수 있게 된다.

873



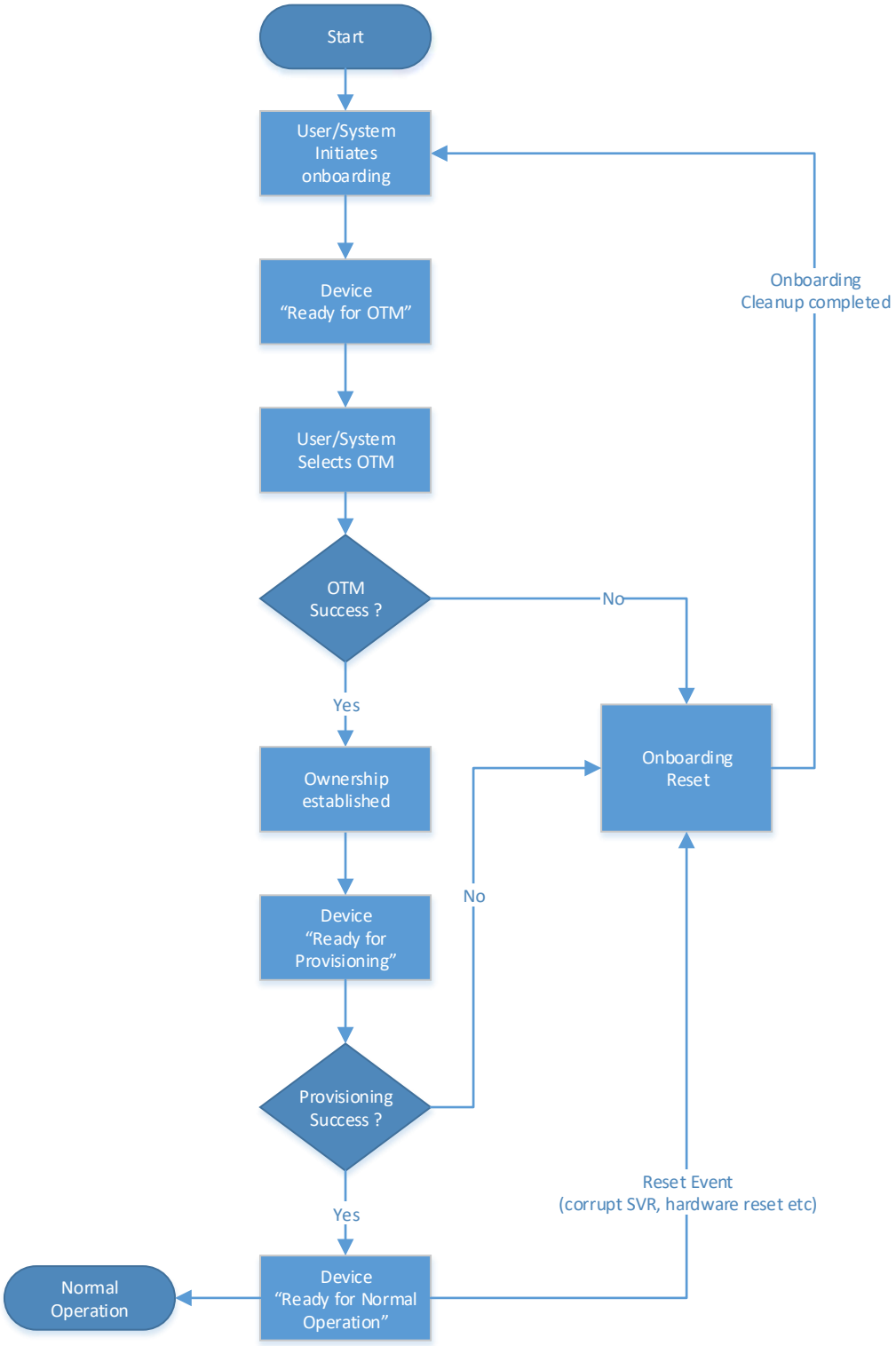
874

도 10 – 온보딩 개요

이 섹션에서는 온보딩과 보안 프로비저닝 절차를 설명하지만, 비 보안 측면의 프로비저닝은 다른 OCF 스펙에서 다룬다. 보안 관점에서 모든 Device 는 OCF 환경에서 다른 Device 와 안전하게 연동/통신할 수 있도록 하기 위한 최소한의 보안 구성을 통해 프로비저닝 되도록 요구된다. 이러한 최소한의 보안 구성은 Onboarded Device "통상 동작 준비 완료"로 정의되며, 이는 섹션 8 에서 규정한다.

5.2.1 온보딩 단계

아래의 흐름도는 온보딩에 관련된 전형적인 단계를 보여준다. 온보딩에는 다양한 비 보안 관련 단계가 포함되지만, 아래의 흐름도는 OCF 환경 내에서 신규 Device 가 기능하도록 하기 위한 보안 관련 구성을 주로 다룬다. 통상적으로 온보딩은 Device 가 정당한 사용자/시스템에 의해 소유되는 것으로 시작해서 작동 환경에 대한 Device 의 구성으로 이어진다. 이러한 구성에는 누가 Device 에 액세스할 수 있고, 어떠한 동작이 수행될 수 있고, 다른 Device 와의 연동을 위해 어떠한 허가를 갖고 있어야 하는지 등의 설정 정보를 포함된다.



890

891 **도 11 – OCF 온보딩 프로세스**

5.2.2 Device 소유자 설정

Device 소유권 설정의 목적은 Device 를 소유/구입한 정당한 사용자가 Device 소유자이며 관리자임을 주장할 수 있도록 하기 위함이다. 이것은 신규 Device 와 OBT 툴 간의 소유권 컨텍스트를 생성하고 Device 의 동작 제어 및 관리를 확인하는 OBT 의 사용을 통해 이루어진다. OBT 는 네트워크 관리 콘솔, device 관리 툴, 네트워크 구축 툴, 네트워크 프로비저닝 툴, 홈 게이트웨이 device, 또는 홈 자동화 컨트롤러와 같은 tool/Server 에 의해 호스트 되는 논리 개체로 간주된다. OBT 를 호스트하는 물리적 device 는 일부 보안 강화 요구 사항의 적용을 받으므로 저장되는 크리덴셜의 무결성 및 기밀성이 보존된다. Device 소유권을 설정하는 tool/Server 를 OBT 라고 한다.

OBT 는 안전하게 Device 소유권을 설정하기 위해 섹션 **Error! Reference source not found.**에 규정된 소유권 이전 방법 중 하나를 사용한다. 소유권 이전이라는 용어가 사용되는 것은 신규 Device 라고 하더라도 Device 의 제조사/제공업자로부터 Device 의 구매자/정당한 사용자에게 소유권이 이전된다고 볼 수 있기 때문이다.

소유권 이전 방법은 Device 의 관리 권한이 부여된 신규 소유자 (OBT 의 사용자)를 설정한다. 소유권 이전에서는 다음과 같은 내용이 설정된다.

- Device 의 /oic/sec/doxm Resource 에서 OBT 에 의해 제공되는 Owner Credential (OC). OC 는 이어지는 상호 작용 시에 Device 와 OBT 가 상호 인증할 수 있도록 한다. OC 는 OBT 의 소유자로 크리덴셜에 기록함으로써 Device 의 사용자/시스템 소유권을 주장한다. OBT 는 소유권 이전의 일부로 Device 아이덴티티도 기록한다.

- Device 소유자는 OTM 을 통해 Device 에 대한 신뢰를 확립한다.
- 경우에 따라 필요한 크리덴셜을 제공함으로써 프로비저닝을 위해 Device 를 준비한다.

5.2.2.1 프로비저닝을 위한 Device 준비

Device 소유권이 설정되면 프로비저닝을 위한 Device 의 준비가 필요하다. 이 과정은 Device 가 bootstrap server (BS)에 접속하고 BS 와 보안 세션을 생성하도록 하는 bootstrapping parameter (BP)를 취득하는 형태로 이루어질 수 있다. 전형적인 부트스트랩 파라미터는 다음과 같다.

- Bootstrap server (BS)/ tool 메타데이터: 이 정보는 bootstrap server 에 액세스하기 위해 사용되는 어드레싱 및 액세스 메커니즘/프로토콜을 포함해야 한다. 어드레싱 정보는 Server 에 접속하기 위해 HTTP 또는 TCP/IP 를 사용하는 경우 Server URI 또는 FQDN 를 포함할 수 있다.

- Bootstrapping credential (BC): 이것은 This Device 가 BS 에 접속하고, BS 에 대해 인증하고, BS 와 보안 세션을 확립하여 BS 로부터 프로비저닝 파라미터를 수신하기 위해 사용해야 하는 크리덴셜이다. OC 의 타입에 따라서는 OC 로부터 BC 를 도출할 수 있다.

OBT 자체가 bootstrap server 로 동작하는 경우, bootstrap server 에 대한 메타데이터는 OBT 자체가 호스트하는 서비스를 가리킬 수 있다. 그러한 시나리오에서는 추가 BC 크리덴셜이 필요하지 않을 수도 있다.

5.2.3 통상적인 동작을 위한 프로비저닝

Device 프로비저닝을 개시하기 위해 필요한 정보를 갖추면, 다음 단계는 Device 가 동작 가능하도록 하는 추가 보안 구성을 프로비저닝하는 것이다. 이 과정은 다양한 파라미터의 설정을 포함할 수 있으며 복수의 단계로 구성될 수도 있다.

예를 들어, Device 가 bootstrapping server 로부터 구성을 수신하는 경우, 프로비저닝은 bootstrap server 에의 연결을 포함해서 구성을 수신할 수 있다. Server 에 의해 호스트된 다양한 Resource 에 대한 ACL 의 프로비저닝도 이 단계에서 수행된다. 단, 프로비저닝은 이 단계에만 제한되는 것이 아님에 주의해야 한다. Device 프로비저닝은 Device 의 작동 생명 주기의 복수의 단계에서 발생할 수 있다. 그러나, Resource 및 Property 상태의 특정 보안 관련 프로비저닝은 보통 이 단계에서 발생하며, 이를 통해 각 Device 는 Onboarded Device "통상 동작 준비 완료" 상태에 이르게 된다. "통상 동작 준비 완료" 상태는 사용되는 특정 OTM 또는 프로비저닝되는 내용의 가변성에 관계 없이 일정하고 명확하게 정의되어야 한다. 단, 개별 OTM 메커니즘 및 프로비저닝 단계는 부가적인 Resource 및 Property 상태를 규정하는 경우가 있다. Device 가 "통상 동작 준비 완료" 상태로 되기 위한 최소 필수 구성은 섹션 8 에서 규정한다.

5.3 프로비저닝

일반적으로 프로비저닝은 Device 가 의도된 환경으로 도입된 후의 프로세스 (온보딩 프로세스의 일부)뿐 아니라 Device 의 제조 및 유통 도중의 프로세스도 포함하는 경우가 있다. 본 스펙에서 보안 프로비저닝은 소유권 이전 중 (소유권 이전 및 온보딩 시의 일부 행위가 Device 에의 일부 데이터의 프로비저닝으로 이어지더라도), bootstrapping 및 provisioning 서비스와의 연동을 위한

크리덴셜의 구성 이후 프로세스, 및 보안 관련 Resource 및 Device 가 나중에 접속할 필요가 있는 서비스의 처리를 위한 크리덴셜의 구성을 포함한다.

소유권 이전이 완료되고 bootstrap 크리덴셜이 설정되면 Device 는 bootstrap server 와 연결하여 적절한 보안 크리덴셜과 파라미터를 프로비저닝 받아야 한다. 이러한 파라미터는 다음을 포함할 수 있다.

- 현재 동일한 OBT 내에 배포된 것으로 간주되는 credential management service (CMS)를 통한 보안 크리덴셜.
- 현재 동일한 OBT 내에 배포된 것으로 간주되지만 향후 AMS 의 일부일 수 있는 ACL provisioning service (APS)를 통한 액세스 제어 정책 및 ACL.

언급한 바와 같이, 확장 가능한 모듈식 설계를 수용하기 위해, 이들 기능은 향후 독립된 server 로 배포될 수 있는 서비스로 간주된다. 현재, 배포는 이들 서비스가 모두 OBT 의 일환으로 배포되는 것을 전제로 하고 있다. 물리적 배포 scenario 에 관계 없이, 동일한 보안 강화 요구 사항이 여기서 논의된 툴 및 보안 프로비저닝 서비스를 호스트하는 모든 물리적 server 에 적용된다.

Device 는 자신의 보안 프로비저닝 상태를 *인지한다*. 자기 인식은 Device 로 하여금 필요 시에 보안 Resource 의 프로비저닝 또는 재 프로비저닝에 대해 적극적으로 대처하여 작동 목표를 달성하도록 한다.

5.3.1 부트스트랩 서비스의 프로비저닝

Device 는 Bootstrap server (BS)를 검색하고 이와 연동하기 위해 요구되는 메타데이터를 포함하는 bootstrap parameters (BP)를 사용해서 검색 또는 프로그램 된다. Device 는 BS 와 안전하게 통신하기 위해 필요한 bootstrap credential (BC)로 구성된다.

Resource 구조에서 /oic/sec/pstat Resource 내의 rowneruuid Property 는 bootstrap service (BSS)를 식별한다.

대칭 키 사용 시에, OC 는 BC 를 도출하는데 사용된다. 그러나, Device 가 소유권 이전 및 그 밖의 프로비저닝 프로세스에 비 대칭 키를 사용할 수 있으면, BC 와 OC 간에 암호 관계가 필요하지 않을 수도 있다.

BC 를 어떻게 생성하는지에 관계 없이, Device 와 BS (및 잠재적으로 그 밖의 servers) 간의 통신은 안전하게 수행되어야 한다. 예를 들어, Device 와의 보안 연결을 위해 PSK 를 사용하는 경우, oic.sec.bss service 는 PSK 가 프로비저닝된 oic.sec.cred Resource 를 포함한다.

5.3.2 그 밖의 서비스의 프로비저닝

잠재적으로 상이한 device 관리 서비스 호스트의 사용을 지원 가능하도록, 각 Device Secure Virtual Resource (SVR)는 관련된 Resource Owner 를 갖는다. DOXS 로도 알려진 온보딩 Device 는 적절한 provider 아이덴티티와 함께 rowneruuid Property 를 프로비저닝 한다.

- Credential Management Service (CMS) : (/oic/sec/cred.rowneruuid)
- Access Manager Service (AMS) : (/oic/sec/acl.rowneruuid and /oic/sec/acl2.rowneruuid)

이러한 서비스가 채워지면, Device 는 적극적으로 프로비저닝을 요청하고 프로비저닝 request 가 허가되었는지 확인한다. 위의 각 service 는 안전하게 수행되어야 하므로 특정 크리덴셜의 프로비저닝을 요구한다. DOXS 서비스는 서비스 프로바이더 Device 로 신호를 보내거나 Device 의 /oic/sec/pstat Resource 의 'tm' Property 에 적절한 벡터를 설정함으로써 위의 서비스를 개시할 수 있다. 이는 Device 로 하여금 크리덴셜 및/또는 액세스 Resource 를 재 프로비저닝하도록 한다.

5.3.3 크리덴셜 프로비저닝

/oic/sec/cred Resource 내에는 몇가지 타입의 크리덴셜을 구성할 수 있다. 현재, 최소한 다음의 크리덴셜이 이에 포함된다: 쌍 대칭 키, 그룹 대칭 키, 인증서, 비 대칭 키, 및 서명된 비 대칭 키. 키는 CMS (예: "oic.sec.cms")에 의해 프로비저닝되거나 Diffie-Hellman 키 합의 프로토콜 또는 기타 수단을 통해 동적으로 프로비저닝될 수 있다.

다음은 보안 연결을 위해 Device 가 PSK 를 업데이트 하는 방법의 예를 보여준다. Device 는, 예를 들어, 보안 연결 시도를 실패하였을 때, 크리덴셜을 업데이트할 필요가 있다고 판단할 수 있다. 그리고 나서, Device 는 CMS 에 크리덴셜의 업데이트를 요청할 필요가 있다. Device 는 크리덴셜 프로비저닝 모드로 전환할 수 있으며 (예: /oic/sec/pstat.Cm=16), 크리덴셜 Resource 의 업데이트를 요청하기 위한 동작 모드를 구성할 수 있다 (예: /oic/sec/pstat.Om="1"). CMS 는 업데이트 요청에 대한 응답으로 신규 쌍 PSK 를 출력한다.

5.3.4 Role 할당 및 프로비저닝

Server 는 보유하고 있는 Resource 에 대한 요청을 수신하면, Resource 를 요청하는 Client 의 role 을 검사하고, 해당하는 role 을 서비스에 대응하는 ACL 에 기술된 제약과 비교한다. 따라서, Client Device 에는 하나 이상의 Role 크리덴셜이 제공되어야 한다.

각 Device 는 크리덴셜 Resource 내에 role 정보를 Property 로 갖고 있다. 따라서, Role 을 프로비저닝 받기 원하는 Client 는 프로비저닝을 탐색하고, 크리덴셜과 ACL 을 프로비저닝 받는 모드로 전환할 수 있다 (이들이 동일한 sever 로부터 제공되는 경우!). 프로비저닝 모드/상태는 일반적으로 /oic/sec/pstat 의 내용에 의해 가리켜진다.

크리덴셜과 ACL 이 프로비저닝되면 Client 는 인증서 role 크리덴셜이 있으면 섹션 10.3.1 에 기술된 대로 role 을 주장할 수 있다.

또는, server 에 client 를 위한 role 정보가 제공되어 있거나 이전에 client 가 server 에 role 을 주장한 적이 있으면, client 는 CoAP payload 를 사용해서 특정 role 을 주장할 수 있다.

예: GET /a/light?roleid={"role":"Role-A"}

client 는 사전에 server 에 어떤 role 이 프로비저닝되어 있는지 알 수 없으며, 일단 동작을 취하고 server 의 응답을 관찰해야 한다. 응답이 허가를 거절하는 것이면 client 는 server 에 해당하는 role 이 프로비저닝되어 있지 않거나 ACL 이 잘못 구성되어 있음을 알게 된다. CoAP payload 에 특정 role 이 지정되어 있지 않으면, 모든 프로비저닝된 role 이 ACL 시행에서 사용된다. Server 가 client 에 대해 제공된 복수의 role 을 갖고 있는 경우, 이들 중 하나의 role 이 허가되면 Resource 에의 액세스가 허가된다.

5.3.5 ACL 프로비저닝

ACL 프로비저닝 시에 Device 는 APS (또는 bootstrap server 가 APS 를 호스트하는 경우, bootstrap server)에의 보안 연결을 확립한다. APS 는 ACL 정책에 따라 Device ACL 을 인스턴스화 또는 업데이트한다.

Device 와 APS 는 ACL 정책에의 변경이 감지되면 Device 에 ACL 프로비저닝이 트리거되어 관찰자 관계를 확립할 수 있다.

APS (예: rt="oic.sec.aps")는 a /oic/sec/sacl Resource 발행의 일부로서 ACL 에 디지털적으로 서명할 수 있다. 서명을 검증하기 위해 Server 가 사용하는 공개키는 크리덴셜 프로비저닝의 일부로 제공될 수 있다. 비 대칭 키 또는 서명된 비 대칭 키 타입의 /oic/sec/cred Resource 를 사용할 수 있다. PublicData Property 는 APS 의 공개키를 포함한다.

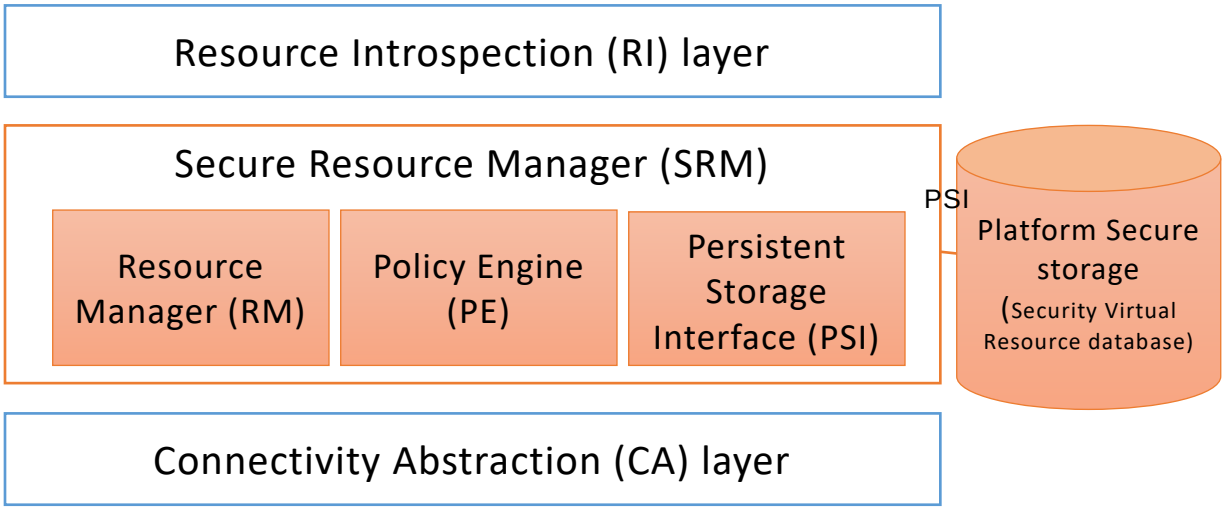
5.4 Secure Resource Manager–(SRM)

SRM 은 전체적인 보안 동작에 있어서 중요한 역할을 담당한다. 요컨대, SRM 은 SVR 의 관리와 Resource 에 액세스하고 이를 다루기 위한 요청에 대한 액세스 제어를 양쪽 다 수행한다. SRM 은 세 개의 주요 기능 요소로 구성된다.

- Resource manager (RM): 1) 필요 시에 지속성 저장장치 (PSI 사용)로부터 SVR 를 로딩하고, 2) 요청된 Resource 와 함께 Policy Engine (PE)을 제공하고, 3) SVR 에 대한 request 에 응답하는 작업을 담당한다. SVR 이 SRM 메모리 내에 저장되어 있는 동안 SVR 은 device 특정 데이터 저장 형식과 일치하는 형식을 갖는다. 그러나, RM 은 저장을 위해 PSI 로 전달되거나 device 외부로 전송되기 전의 SVR 데이터 구조를 제어하는데 JSON 형식을 사용한다.

- Policy Engine (PE): SVR 에 액세스하기 위한 request 를 수신하고, 액세스 제어 정책에 기반하여 "ACCESS_GRANTED" 또는 "ACCESS_DENIED"로 request 에 응답한다. 액세스 결정을 하기 위해, PE 는 적절한 ACL 을 참조하여 DTLS 에 의해 인증된 subject (Device 또는 role)에게 주어진 request 를 처리할 수 있는 최적의 Access Control Entry (ACE)를 찾는다.

- Persistent Storage Interface (PSI): PSI 는 자신의 메모리 및 저장장치 내의 파일을 처리하기 위해 RM 을 위한 API 의 집합을 제공한다. SRM 은 가능한 경우 Platform 의 보안 실행 환경 내에 구현될 수 있도록 모듈 식으로 디자인된다.



도 12 – OCF SRM 아키텍처

5.5 크리덴셜 개요

Device 는 양방향 통신에서 상호 간에 아이덴티티와 role 을 증명하기 위해 크리덴셜을 사용할 수 있다. 크리덴셜은 대칭 또는 비 대칭일 수 있다. 각 device 는 OBT 또는 CMS 에 의해 프로비저닝된 다른 device 의 크리덴셜과 더불어 해당하는 경우 자시의 크리덴셜의 비밀 및 공개 부분을 저장한다. 이러한 크리덴셜은 참여하는 당사자의 아이덴티티를 검증하기 위한 보안 통신 세션 (예: DTLS

사용)을 확립하는데 사용된다. 인증된 세션이 확립되면 Role 크리덴셜은 device 의 하나 이상의 role 을 주장하는데 사용된다.

6 Discovery 프로세스를 위한 보안

discovery 메커니즘의 주요 기능은 Server 가 갖고 있는 Resource 에 대해 Resource 에 관한 속성과 가능한 이후의 link 관계와 더불어 Universal Resource Identifier (URI, link 라고 칭함)를 제공하는 것이다. (다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification 의 섹션 10 에 따름)

6.1 Discovery 를 위한 보안 고려 사항

discovery 프로세스를 정의할 때, 민감한 정보에 대한 보안 또는 애플리케이션의 프라이버시 요구를 위반하지 않고 discovery 를 수행하는 개체에 대해 최소한의 Resource 집합만을 노출시켜야 함에 주의해야 한다. 여기에는 대응하는 메타데이터뿐 아니라 Resource 에 포함된 데이터도 포함된다.

연장성 및 확장성을 확보하기 위해 본 스펙에서는 각각의 개별 Resource 의 발견 가능성에 대해 어떠한 필수 요건도 정하지 않는다. 대신에, Resource 를 보유하고 있는 Server 는 각 Resource 에 대해 ACL 에 의존해서 요청자 (Client)가 Resource 를 보거나 다룰 수 있도록 인가되었는지를 판단한다.

/oic/sec/acl2 Resource 는 Server 가 보유한 Resource 에의 액세스를 관장하는 ACL entry 를 포함한다. (섹션 13.4 참조)

ACL 의 관점에서 프라이버시와 Resource 의 발견 가능성과는 별도로, discovery 프로세스는 그 자체가 보안성을 가져야 한다. 본 스펙에서는 discovery 프로세스에 대해 다음과 같은 요구 사항을 설정한다.

1. 발견된 Resource 에 대해 무결성 보호를 제공한다.
2. 민감하다고 여겨지는 발견된 Resource 에 대해 기밀성 보호를 제공한다.

Resource 의 발견은 알려진 Resource "/oic/res"에 대해 RETRIEVE 동작 (unicast 또는 multicast)을 수행함으로써 이루어진다.

discovery request 가 비 보안 채널 (DTLS 를 사용하지 않는 멀티캐스트 또는 유니캐스트)로 전송되면, Server 가 requester 의 아이덴티티를 결정할 수 없게 된다. 그러한 경우에 응답 전에 Client 를 인증하고자 하는 Server 는 비 보안 /oic/res 응답 내에 보안 discovery URI (예: coaps://IP:PORT/oic/res)를 나열할 수 있다. 이는 보안 discovery URI 가 모든 Client 에 의해 디폴트로 발견 가능함을 의미한다. 그 후, Client 는 보안 discovery URI 에 DTLS 를 사용해서 별도의 unicast 요청을 전송해야 한다.

보안 발견의 경우, Client 가 CRUDN 동작 중 최소한 하나를 수행하도록 허가되었을 때만 (즉, CRUDN 플래그의 비트간 OR 값이 참이어야 한다) 관련된 ACL2 를 갖는 모든 Resource 가 /oic/res 에 대한 응답 내에 나열된다.

예를 들어, 아래와 같이 DeviceId "d1"의 Client 가 아래와 같은 ACL2 를 갖는 DeviceId "d3"의 Server 상의 "/door" Resource 에 대해 RETRIEVE request 동작을 수행한다.

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



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

ACL 은 Client "d1"이 Resource 에 대한 RETRIEVE 허가를 가진 것을 나타낸다. 그러므로, device "d1"이 Server "d3"의 /oic/res Resource 에 대해 discovery 를 수행하면 이에 대한 응답은 "/door" Resource 메타데이터의 URI 를 포함한다. Client "d2"는 양쪽의 Resource 에 대한 액세스 권한을 갖게 된다. ACE2 는 "d4"의 업데이트를 방지한다.

보안 Interface 로부터 d1 으로 전달되는 d3 의 /oic/res Resource 에 대한 발견 결과:

```
[
  {
    "href": "/door",
    "rt": ["oic.r.door"],
    "if": ["oic.if.b", "oic.ll"],
      "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
  }
]
```

보안 Interface 로부터 d2 로 전달되는 d3 의 /oic/res Resource 에 대한 발견 결과:


```
[
  {
    "href": "/door",
    "rt": ["oic.r.door"],
    "if": ["oic.if.b", "oic.ll"],
      "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
  },
  {
    "href": "/door/lock",
    "rt": ["oic.r.lock"],
    "if": ["oic.if.b"],
    "type": ["application/json", "application/exi+xml"]
  }
]
```

보안 Interface 로부터 d4 로 전달되는 d3 의 /oic/res Resource 에 대한 발견 결과:

```
[
  {
    "href": "/door/lock",
    "rt": ["oic.r.lock"],
    "if": ["oic.if.b"],
    "type": ["application/json", "application/exi+xml"],
    "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
  }
]
```

비 보안 Interface 로부터 모든 device 로 전달되는 d3 의 /oic/res Resource 에 대한 발견 결과:

```
[
  {
    "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
    "href": "/light",
    "rt": ["oic.r.light"],
    "if": ["oic.if.s"]
  }
]
```


1199

7 보안 프로비저닝

7.1 Device 아이덴티티

논리적 device 인 각 Device 는 device ID 로 식별된다.

Device 는 device 온보딩의 일부로 설정되는 Device ID 값에 의해 식별되어야 한다. /oic/sec/doxm Resource 는 Device ID 형식 (예: urn:uuid)를 지정한다. Device ID 는 해당하는 OCF network 의 동작 범위 내에서 고유해야 하며, 범용적으로 고유한 것이 좋다. 네트워크 내에서의 Device ID 의 고유성은 device 온보딩 시에 확보되어야 한다. Device OBT 는 선택된 신규 device 식별자가 기존에 네트워크에 도입된 다른 device 와 충돌하지 않는지 확인해야 한다.

Device 는 Device ID 와 /oic/sec/cred Resource 를 사용한 암호화 크리덴셜의 관계를 유지한다. peer device 의 인증 크리덴셜을 확인할 때, Device 는 the /oic/sec/cred Resource 가 권한이 있는 것으로 간주한다.

Device 는 /oic/sec/doxm Resource 내의 device ID 를 유지한다. Device 는 /oic/sec/cred Resource 내의 자신과 다른 device 둘 다의 크리덴셜 목록을 유지한다. device ID 는 device 자신의 크리덴셜과 다른 device 의 크리덴셜을 구별하는데 사용될 수 있다. 더욱이, /oic/sec/cred Resource 는 device 에 대해 복수의 크리덴셜을 포함할 수 있다.

Device ID 는:

- 고유해야 하고,
- 불변이어야 하며,
- 증명 가능해야 한다.

제조사 인증서 사용 시에는 이 섹션의 후반에 설명하는 바와 같이 인증서가 ID 를 device 내의 저장된 비밀에 바인딩하는 것이 좋다.

OCF 스펙에서 Platform 으로 불리는 물리적 device 는 복수의 Device 를 호스트할 수 있다 Platform 은 Platform ID 로 식별된다. Platform ID 는 전역적으로 고유해야 하며 device 내에 무결성이 보호되도록 삽입되어야 한다 (예: 내부 보안 저장 또는 서명 및 인증).

주: OCF Platform 은 고유한 식별자 및 비밀 보장을 위해 사용되는 보안 실행 환경을 가질 수 있다. Platform 이 복수의 device 를 호스트하는 경우에는, 각각의 Device 에 적절한 별도의 보안을 제공하기 위해 소정의 메커니즘이 필요하게 된다.

7.1.1 UAID 를 갖는 Device 용 Device 아이덴티티

OCF root CA (섹션 **Error! Reference source not found.**에 규정된)에 연쇄된 인증서와 함께 제조사 인증서를 사용할 때는, 제조사가 인증서 주체 CN 필드 내에 Platform ID 를 포함해야 한다. 그러한 경우에, device ID 는 이 섹션에 설명된 Unique Authenticable IDentifier (UAID) 기법에 따라 생성할 수 있다.

Device 를 식별하고 보호하기 위해, Platform Secure Execution Environment (SEE)는 호스팅하는 각각의 Device 에 대해 신규 Dynamic Public Key Pair (DPKP)를 생성하거나 단순히 제조사에 의해 내장된 동일한 공개 키 크리덴셜 (Embedded Platform Credential (EPC))를 사용할 수 있다. 어떠한 경우에도, Platform SEE 는 각각의 Device 에 대한 device 아이덴티티를 생성하기 위해 Random Number Generator (RNG)를 사용한다. UAID 는 eitherEPC 만을 사용해서 생성되거나 DPC 와 EPC 가 사용 가능한 경우 이 둘의 조합을 사용해서 생성된다. 둘 다 사용 가능할 때, Platform 은 이 섹션에 설명된 바와 같이 양쪽의 key pair 를 사용해서 UAID 를 생성해야 한다.

Device ID 는 device 의 공개 키 및 관련된 OCF Cipher Suite 로부터 형성된다. Device ID 는 다음과 같은 과정을 통해 형성된다.

1. Dynamic Public Key 의 OCF Cipher Suite 를 결정한다. The Cipher Suite 커브는 Device 보안 메커니즘과의 사용을 의도한대로 SubjectPublicKeyInfo 내에 사용되는 AlgorithmIdentifier 의 용법과 일치해야 한다. CipherSuite 의 인코딩은 다음의 계산에서 'csid' 값으로 사용한다. Dynamic Public key 를 위한 OCF Cipher Suite 가 Platform 인증서 (EPC) 내에 가리켜진 ciphersuite 와 다를 때는 OCF Cipher Suite 가 사용됨에 주의하기 바란다.

2. EPC 로부터 내장된 공개 키의 값을 추출한다. 추출된 값은 인증서의 SubjectPublicKeyInfo 내에 정의된 subjectPublicKey 의 값에 대응해야 한다. 이하, 이것을 EPK 라고 한다. 인증서로부터 공개 키가 추출되면, AlgorithmIdentifier 가 인증서 내의 CipherSuite 에 대한 예측값과 일치하는지 확인한다.

3. DPC 로부터 공개키의 값을 추출한다. 추출된 값은 SubjectPublicKeyInfo 내에 정의된 subjectPublicKey 의 값에 대응해야 한다. 이하, 이것을 DPK 라고 한다.

4. Cipher Suite 용 해시를 사용해서 다음을 계산한다.
   h = hash('uaid' | csid | EPK| DPK | <other_info>)

Other_info 는 1) /oic/d 내에 가리켜진 device type (읽기 전용으로 제조사에 의해 설정될 수 있음), 2) 두 개의 공개 키 쌍이 존재하는 경우 (하나는 내장되고, 다른 하나는 동적으로 생성되는), 양쪽 공개 키가 모두 포함된다.

5. h 의 처음 128 비트를 취해서 자른다.
   UAID = h[0:16] # leftmost 16 octets

6. 다음과 같이 바이너리 UAID 를 ASCII 스트링으로 변환한다.
   USID = base27encode(UAID)

```
def base_N_encode(octets, alphabet):
long_int = string_to_int( octets )
    text_out = ''
    while long_int > 0:
        long_int, remainder = divmod(long_int, len(alphabet))
        text_out = alphabet[remainder] + text_out
    return text_out

b27chars = 'ABCDEFGHJKMNPQRTWXYZ2346789'
def b27encode(octet_string):
    """Encode a octet string using 27 characters. """
    return base_N_encode(octet_string, _b27chars )
```

7. 'urn:usid:'에 USID 의 스트링 값을 뒤에 붙여 Device ID 의 최종 스트링 값을 형성한다.
   urn:usid:ABXW….

공개 키의 인코딩에는 SubjectPublicKeyInfo 용으로 RFC 7250 에 기술된 형식을 사용해야 한다.

7.1.1.1 **UAID 의 검증**

새로 생성된 Device ID (UAID)와 공개 키 쌍 (DPC)을 사용 가능하도록 하려면 device Platform 은 내장된 개인 키 (제조사 내장 공개 키 및 인증서에 대응)를 사용해서 그것 (Platform)이 실제로 DPC 와 UAID 를 생성하여 DPC 의 사용 책임을 신규 device 소유자에게 일임하였음을 보증하는 토큰에 서명해야 한다. 이는 또한 신규 DPC 와 UAID 에서의 사용을 위해 생태계가 제조사 인증서로부터 device 발행 인증서로 신뢰성을 확장할 수 있도록 한다. 신뢰성의 정도는 device SEE 의 강화 레벨에 의존한다.

Dev_Token=Info, Signature(hash(info))

Signature algorithm=ECDSA (EPC 내 또는 DPC 에 가능한 것과 동일한 알고리즘일 수 있다)

Hash algorithm=SHA256

Info=UAID| <Platform ID> | UAID_generation_data | validity

UAID_generation_data=UAID 생성에 사용된 해시 알고리즘으로 전달된 데이터.

Validity=일 단위의 유효 기간 (토큰이 유효한 기간)

7.2 Device 소유권

이 섹션은 참고 섹션이다. Device 는 암호화 크리덴셜을 사용해서 인증 가능한 아이덴티티를 갖는 보안 종단 점인 논리적 개체이다. Device 가 처음 초기화되었을 때는 소유자가 없는 상태이다. device 소유권의 설정은 이에 의해 device 가 OBT 에 대해 자신의 아이덴티티를 주장하고 OBT 가 device 에 대해 자신의 아이덴티티를 주장하는 프로세스이다. 이 교환은 device 내에서 소유권 상태를 변경하게 되므로 다른 OBT 가 device 에 대해 관리 제어를 주장하는 것을 방지하게 된다.

소유권 이전 프로세스는 OBT 가 신규 device 의 /oic/sec/doxm Resource 의 "Owned" Property 검사를 통해 소유자가 없는 상태의 신규 device 를 탐색하는 것에서 시작한다. 소유권 이전은 최종적으로 다음과 같은 결과를 얻는다.

1. 신규 device 와 OBT 간의 보안 세션 확립.
2. 다음 중 하나를 선택적으로 주장한다.
    a. Platform 에의 OBT 의 근접 (PIN 사용).
    b. Platform 의 공급자, 모델, 및 그 밖의 Platform 고유 속성을 주장하는 제조사의 인증서.
3. device 식별자 결정.
4. device 소유자 결정.
5. device 소유자 지정 (예: OBT 의 Device ID).
6. Device 에 소유자의 크리덴셜을 프로비저닝.

7. 신규 device 의 'Owned" 상태를 TRUE 로 설정.

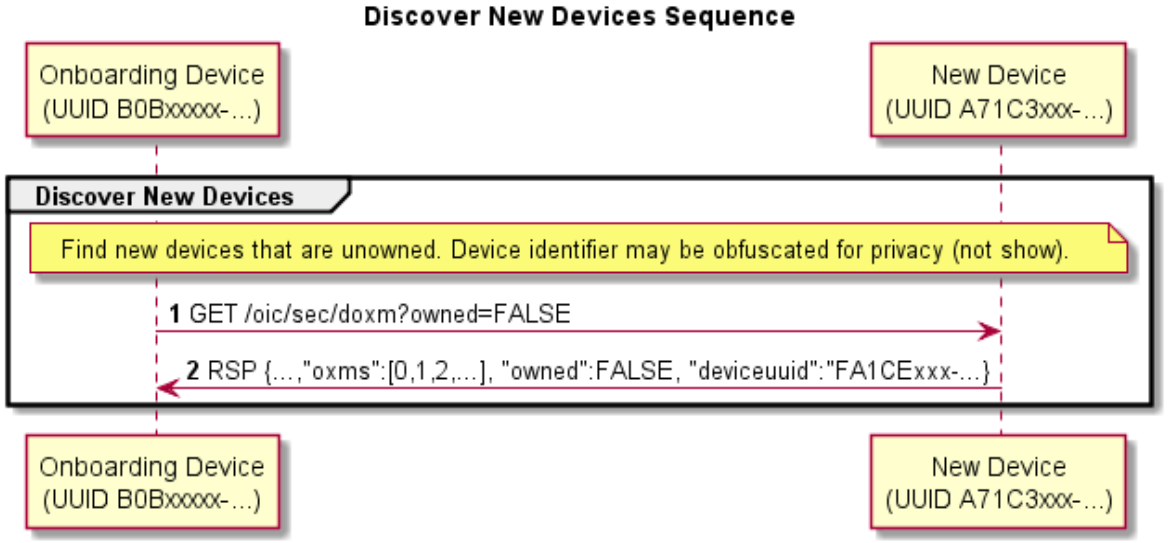
7.3 Device 소유권 이전 방법

7.3.1 OTM 구현 요구 사항

본 문서는 소유권 이전을 위한 몇 가지 방법에 관한 스펙을 제공한다. 각각의 소유권 이전 방법은 선택적으로 구현할 수 있다. 단, 각 device 는 공급자 특정 방법을 제외한 소유권 이전 방법 중 최소한 하나를 구현해야 한다.

본 문서에 포함된 모든 소유권 이전 방법 (OTM)은 선택적으로 구현할 수 있다. 각 공급자는 본 스펙에서 규정된 최소한 하나의 OTM 을 선택해서 구현해야 한다. 그러나, OCF 는 공급자 특정 접근 방식도 존재할 것으로 예상한다. 공급자가 공급자 특정 소유권 이전 방법과 다른 공급자로부터의 OBT 간에 상호 운용성을 갖고자 할 때는, 공급자가 OBT 공급자와의 직접적인 협력을 통해 상호 운용성을 획득해야 한다. 그렇지만 OTM 의 표준화가 바람직한 접근법이라고 할 수 있다. 그러한 경우에 공급자가 공급자 특정 OTM 을 설계할 수 있도록 하기 위해 아래에 일련의 가이드라인을 제공한다. (섹션 **Error! Reference source not found.** 참조).

device 소유권 이전 방법 (doxm) Resource 는 공급자 특정 방법을 수용하기 위해 확장 가능하다. 모든 OTM 방법은 OBT 가 device 의 역량 제약 내에서 주어진 신규 device 에 대해 가장 적합한 OC 를 결정할 수 있도록 해야 한다. OBT 는 신규 device 가 지원하는 크리덴셜 타입을 확인하고 OBT 가 device 제약 내에서 크리덴셜 타입을 선택할 수 있도록 한다.



도 13 - 신규 Device 시퀀스 검색

단계	설명
1	OBT 가 신규 device 가 아직 소유자가 없는 상태인지 확인한다.
2	신규 device 가 소유권 상태와 지원하는 소유권 이전 방법을 포함하는 /oic/sec/doxm Resource 를 리턴한다. 이는 성공적인 소유권 이전에 이어서 변경될 수 있는 임시 device ID 도 포함한다. device 는 guest device 로서의 discovery 가 용이하도록 임시 ID 를 제공하는 것이 좋다. 섹션 Error! Reference source not found.는 소유권 이전 방법 선택 시의 보안 고려 사항을 제공한다.

표 2 - 신규 Device 상세 검색

공급자 특정 device 소유권 이전 방법은 공급자 특정 device 소유권 이전으로부터 기인하는 OC 에 대한 /oic/sec/doxm Resource 스펙을 고수해야 한다. 공급자 특정 방법은 OBT 에 의한 신규 device 에서의 신뢰성 확립 및 신규 device 에 의한 OBT 에서의 선택적인 신뢰성 확립을 위한 프로비전을 포함하는 것이 좋다.

공급자 특정 소유권 이전 방법의 최종 상태는 신규 device 가 OBT 를 인증하고 OBT 가 신규 device 를 인증할 수 있도록 한다.

확립된 세션을 사용한 성공적인 소유권 이전에 이어서 추가적인 프로비저닝 단계가 적용되는 경우가 있지만, 그러한 프로비저닝 단계는 기술적으로 OBT 가 예측하지 않을 수 있는 프로비저닝 단계로 간주되므로 OBT 프로비저닝에 의해 무효화되는 경우가 있다.

7.3.2 SharedKey 크리덴셜 산출

SharedKey 크리덴셜은 온보딩에 사용되는 DTLS 핸드세이크로부터 기인하는 key_block 값을 수신하는 PRF 를 사용해서 도출된다. Server 와 Device OBT 는 공급자 제품 전반에 걸쳐서 상호 운용성을 확보하기 위해 다음의 계산을 사용한다.

SharedKey = *PRF*(Secret, Message);

여기서,

- PRF 는 RFC5246 섹션 5 에 정의된 TLS 1.2 PRF 를 사용해야 하고,
- Secret 는 DTLS handshake 로부터 기인하는 key_block 이고,
  - RFC5246 섹션 6.3 참조
  - key_block 의 길이는 cipher suite 에 의존

1359 • (예: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 의 경우 96 바이트,
1360 TLS_PSK_WITH_AES_128_CCM_8 의 경우 40 바이트)

1361 - Message 는 다음의 접합이며,

1362 ▪ 현재 온보딩 방법을 위한 DoxmType 스트링 (예: "oic.sec.doxm.jw")

1363 • 특정 DoxmType 에 대해서는 "섹션 13.1.1 OCF 정의 소유권 이전 방법" 참조

1364 ▪ OwnerID 는 device 소유자 식별자와 SharedKe 를 유지하는 devic 를 식별하기 위한 UUID 이다.

1365 • RFC4122 섹션 4.1.2 에 규정된 바와 같이 raw 바이트를 사용한다.

1366 ▪ Device ID 는 신규 device 의 UUID Device ID 이다.

1367 • RFC4122 section 4.1.2 에 규정된 바와 같이 raw 바이트를 사용한다.

1368 - SharedKey Length 는 32 octet 이다.

1369 ▪ 이후의 DTLS 세션이 128 비트 암호화 cipher suite 를 사용하면 왼쪽의 16 octet 이 사용된다.
1370 256 비트 암호화 cipher suite 를 사용하는 DTLS 세션은 32 octet 전부를 사용한다.

1371 ### 7.3.3 인증서 크리덴셜 생성

1372 인증서 크리덴셜은 보안 양방향 통신을 위해 Device 에 의해 사용된다. 인증서는 CMS 또는 외부
1373 인증기관 (CA)에 의해 발행된다. 이 CA 는 상호간에 Device 의 상호 인증을 위해 사용된다. 인증서
1374 생성을 위한 온보딩 세부 사항은 본 스펙의 이후 버전에서 규정한다.

1375

1376 ### 7.3.4 Just-Works 소유권 이전 방법

1377 Just-works 소유권 이전 방법은 소유자의 네트워크 내에서 device 의 프로비저닝을 위한 보안
1378 연결의 확립에 사용되는 사전 공유 키인 대칭 키 크리덴셜을 생성한다. 추가적인 크리덴셜 및
1379 Resource 의 프로비저닝은 소유권 설정에 이어지는 일반적인 단계이다. 이와 같은 사전 공유 키를
1380 SharedKey 라고 한다.

1381 소유권 이전 프로세스는 OBT 가 신규 device 에 의해 호스트되는 Device 에서 /oic/sec/doxm
1382 Resource 의 "owned" Property 검사를 통해 소유자가 없는 상태의 신규 device 를 탐색하는 것에서
1383 시작한다.

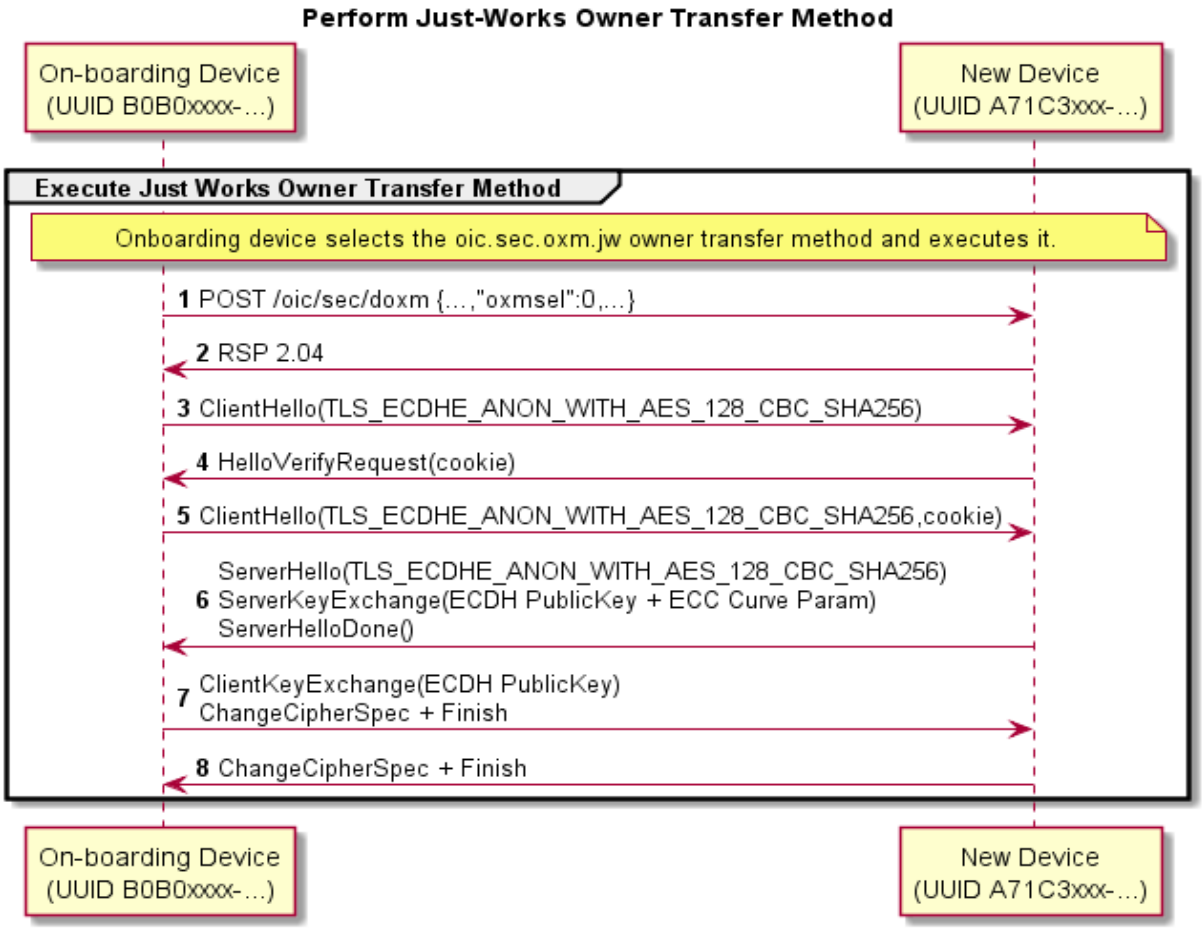
1384 OBT 가 device 의 소유되지 않은 상태를 확인하면, Just-works 소유권 이전 방법 수행 시에, OBT 는
1385 키 합의 프로토콜로 anonymous Elliptic Curve Diffie-Hellman (ECDH)가 사용되는 DTLS 키 교환
1386 프로세스에 의존한다.

1387 다음과 같은 OCF 정의 공급자 특정 ciphersuite 가 Just-works 소유권 이전 방법에 사용된다.

1388

1389 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256,
1390 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256

1391 이들은 IANA 에 등록되어 있지 않아서, ciphersuite 값은 사적인 사용을 위해 예약된 영역 (0xFF00
1392 ~ 0xFFFF)으로부터 할당된다. 할당되는 값은 각각 0xFF00 과 0xFF01 이다.



1393
1394 **도 14 –Just-Works 소유권 이전 방법**

단계	설명
1, 2	OBT 가 Device 에 'Just Works' 방법을 선택하였음을 통지한다.
3 - 8	익명 Diffie-Hellman 을 사용해서 DTLS 세션이 확립된다. 주: 이 방법은 사용자가 중간자 공격의 잠재적인 위험을 인지하고 무결 네트워크 내에서 방법을 수행하기 위한 사전 주의를 취했음을 가정한다.

표 3 – Just-Works 소유권 이전 방법 상세

7.3.4.1 보안 고려 사항

익명 Diffie-Hellman 키 합의는 중간 공격자에게 취약하다. 이 방법의 사용에 있어서는 OBT 와 신규 device 가 둘 다 공격하는 device 가 없는 비교적 안전한 환경에서의 온보딩을 전제로 해서 'just-works' 방법을 수행하는 것으로 가정한다.

이 방법은 확인된 device ID 가 device 에 확실하게 할당되어 있는지 증명할 수 있는 신뢰성 있는 방법이 아니다.

신규 device 는 소유된 device 로 전환되기 전에 프라이버시 관련 추적을 방지하기 위해 guest device 로 취급되는 동안 임시 device ID 를 사용해야 한다. device 는 소유권 이전이 발생하는 보안 세션 동안에 임시 값과 다를 수 있는 영구적인 device ID 를 주장해야 한다. OBT 는 주장된 Device ID 가 기존에 사용되고 있는 Device ID 와 충돌하지 않는지 확인한다. Device ID 가 이미 사용되고 있는 것이라면 기존의 크리덴셜을 사용해서 보안 세션을 확립한다.

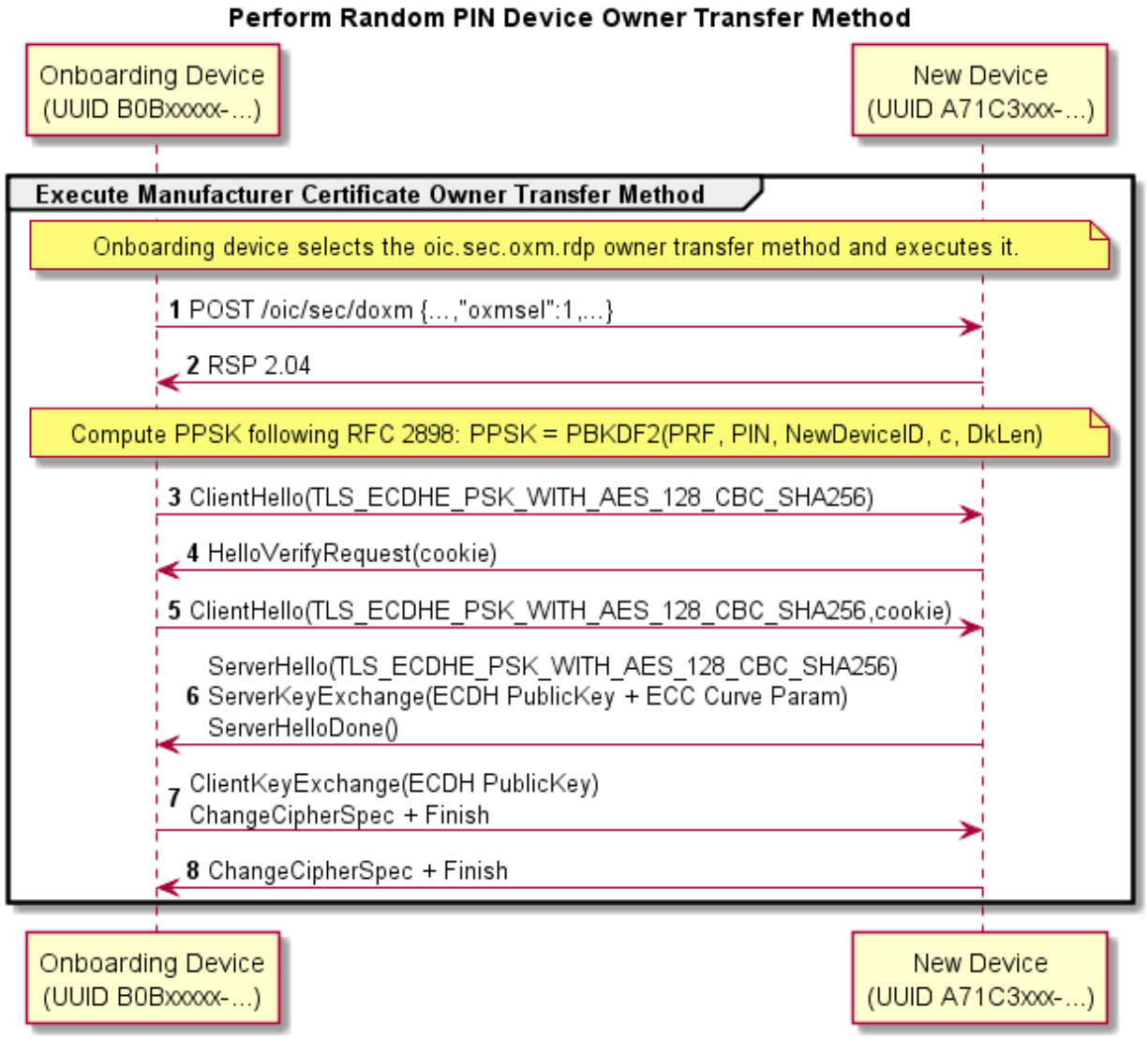
확립된 device 크리덴셜을 가진 소유되지 않은 Device 는 손상된 device 를 가리키는 것일 수 있다.

7.3.5 Random PIN 기반 소유권 이전 방법

Random PIN 방법은 신규 device 와 OBT 간의 물리적 근접을 확립하여 중간자 공격을 방지할 수 있다. Device 는 대역 외 채널을 통해 OBT 로 전달되는 난수를 생성한다. 대역 외 통신 채널이란 device 소유권 이전 방법의 정의 범위를 벗어나는 것을 의미한다.

OBT 와 신규 Device 는 대역 외 채널을 통해 신규 Device 에 물리적으로 액세스함으로써 누구가가 소유권 이전을 인가하였다는 증거로 키 교환 시에 PIN 을 사용한다.

1418 #### 7.3.5.1 Random PIN 소유권 이전 시퀀스



1419

1420 **도 15 – Random PIN 기반 소유권 이전 방법**

단계	설명
1, 2	OBT 가 Device 에 'Random PIN' 방법을 선택하였음을 통지한다.
3 - 8	PSK 기반 Diffie-Hellman ciphersuite 을 사용해서 DTLS 세션이 확립된다. PIN 이 PSK 파라미터로 제공된다. PIN 은 신규 device 에 의해 무작위로 생성되어 신규 device 와 OBT 간에 근접 컨텍스트를 확립하는 대역 외 채널을 통해 전달된다. 보안 원칙은 공격하는 device 가 충분히 근접하지 않아서 PIN 을 가로채지 못하도록 하는 것이다.

표 4 – Random PIN 기반 소유권 이전 방법 상세

random PIN 기반 device 소유권 이전 방법은 RFC2898 에 의해 정의된 의사 난수 생성 기능 (PBKDF2) 및 대역 외 방식을 통한 PIN 교환을 사용하여 사전 공유 키를 생성한다. PIN 인증 사전 공유 키 (PPSK)는 PSK 를 받아들이는 TLS ciphersuite 로 제공된다.

PPSK = PBKDF2(PRF, PIN, Device ID, c, dkLen)

PBKDF2 기능은 다음과 같은 파라미터를 갖는다.

- PRF –RFC5246 에 의해 정의된 TLS 1.2 PRF 를 사용한다.
- PIN – 대역 외 채널을 통해 취득.
- Device ID – 신규 device 의 UUID.
    - RFC4122 섹션 4.1.2 에 규정된 raw 바이트를 사용한다.
- c –1000 으로 초기화된 반복 횟수
- dkLen –도출된 PSK 의 octet 단위의 원하는 길이.

7.3.5.2 보안 고려 사항

Random PIN 메커니즘의 보안은 PIN 의 엔트로피에 의존한다. 엔트로피가 충분하지 않은 PIN 을 사용하면 중간 공격자가 온보딩의 일부로 프로비저닝된 장기적인 크리덴셜을 획득하게 되는 경우가 있다. 특히, 프로비저닝된 대칭 키 크리덴셜을 학습하면 공격자가 온보딩된 device 로 가장할 수 있게 된다.

온라인 무차별 공격에 대응하기 위해서는 PIN 의 엔트로피를 40 비트 이상으로 하는 것이 좋다. 12 자리의 숫자 PIN, 8 문자의 영숫자 (0-9a-z), 또는 7 문자의 대소문자 구별 영숫자 PIN (0-9a-zA-Z)를 예로 들 수 있다. 공격자가 네트워크 상에서 활동 상태일 때 중간자 공격 (MITM)을 통해 OBT 와 device 간의 메시지를 가로채서 수정할 수 있다. MITM 에서, 공격자는 실시간으로, 즉 peer

타임아웃 전에, 키 교환 메시지로부터 PIN 을 획득하여 연결 시도를 중단해야 한다. PIN 을 획득하면, 공격자는 키 교환의 인증 단계를 완료할 수 있다.

여기에 제공된 지침은 40 비트의 엔트로피를 요구하지만, 이에 대한 보증은 공격자가 사용 가능한 resource 에 달려있다. 무차별 추측 공격은 본질적으로 병렬화 가능하므로 더 많은 core/thread 가 추가될수록 공격은 선형적으로 증가하게 된다. 따라서, 더 보수적으로 엔트로피의 양을 64 비트로 잡을 수 있다. Random PIN OTM 은 ECDHE 키 교환을 포함하는 DTLS ciphersuite 의 사용을 필요로 하므로, Random PIN OTM 의 보안은 항상 최소한 JustWorks OTM 의 보안과 동등하다.

Random PIN OTM 에는 PBKDF2 를 사용해서 PIN 으로부터 키 재료를 도출하는 옵션도 있다. 이에 대한 논리적 근거는 조절 가능한 양 (PBKDF2 반복 횟수)에 의해 공격의 추측마다 소요되는 비용을 증가시킴으로써 무차별 공격의 비용을 증가시키는 것이다. 이론상 이것은 PIN 의 엔트로피 요구 조건을 완화시키는 효과적인 방법이다. 하지만 유감스럽게도 이것은 정직한 peer 에 의해 소요되는 X 배의 시간 증가가 공격자의 X 배 시간 증가로 직접 변환되지 않으므로 감소를 정량화하기 어려운 단점이 있다. 이러한 불균형은 공격자가 정직한 peer 에게 제공되지 않는 특수한 구현 및 하드웨어를 사용할 수 있음에 기인한다. 이러한 이유로, PIN 에 대해 얼마만큼의 엔트로피를 사용할지를 결정할 때, 구현자는 PBKDF2 가 아무런 보안도 제공하지 않는다는 전제 하에 PIN 이 충분한 엔트로피를 갖도록 하는 것이 좋다.

Random PIN device 소유권 이전 방법 보안은 랜덤하게 생성된 PIN 을 신규 device 로부터 OBT 로 전달하기 위한 보안 대역 외 방법이 존재한다는 가정에 의존한다. OOB 채널이 PIN 의 일부 또는 전부를 공격자에게 누출하면, 이것은 PIN 의 엔트로피를 감소시켜서 위에 설명한 공격이 성립하게 된다. 대역 외 메커니즘은 OBT 와 신규 device 간의 근접이 필요하도록 선택하는 것이 좋다. 공격자가 대역 외 채널에는 침입하지 않는 것으로 가정한다. OOB 채널의 예로, device 는 OBT 소프트웨어에 입력할 PIN 을 표시할 수 있다. 다른 예로, device 에서 PIN 을 2D 바코드로 인코딩해서 표시하고 OBT device 에서 카메라로 캡쳐해서 디코딩하는 방법이 있다.

7.3.6 제조사 인증서 기반 소유권 이전 방법

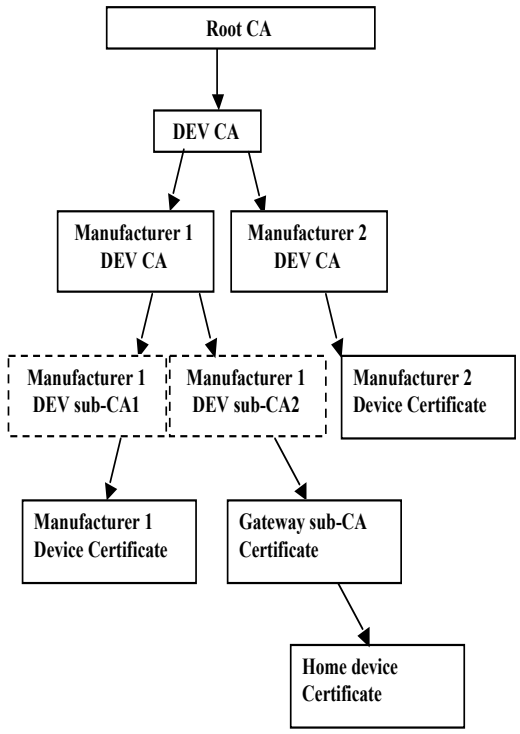
제조사 인증서 기반 소유권 이전 방법은 제조사에 의해 device 에 내장된 인증서를 사용해야 하고 서명된 OBT 를 사용할 수 있으며, 이는 device 와 OBT 간의 Trust Anchor 를 결정한다.

인증서 기반 소유권 이전 사용 시에, device 는 사용자의 네트워크 상에서 신규 device 가 동작하도록 하는 프로세스에서 OBT 로 device 의 아이덴티티를 인증하기 위한 인증서 데이터를 가진 비 대칭 키를 사용해야 한다. 온보딩 프로세스는 다음과 같은 개별 단계를 포함한다.

1) Pre-on-board 조건
    a. 제조사 인증서를 포함하는 Device 의 크리덴셜 Resource (/oic/sec/cred)의 크리덴셜 요소는 다음의 속성에 의해 식별되어야 한다.
        i. subject Property 가 Device 를 가리켜야 한다.
        ii. credusage Property 가 크리덴셜이 제조사 인증서를 포함하는 것을 가리키는 스트링 "oic.sec.cred.mfgcert"을 포함해야 한다.
    b. 제조사 인증서 체인은 Trust Anchor 를 포함하는 optionaldata Property 와 함께 식별된 크리덴셜 요소의 publicdata Property 에 포함되어야 한다.
    c. device 는 고유하고 불변의 ECC 비 대칭 키 쌍을 포함해야 한다.
    d. device 가 소유권 이전의 일부로 OBT 의 인증을 요구할 때는, OBT 가 등록되어 있고 소정의 Trust Anchor 에 의해 서명된 고유하고 불변의 ECC 비 대칭 키 쌍을 위한 인증서를 취득하고 있는 것으로 간주한다.
    e. 사용자는 네트워크 액세스 정보와 계정 정보 (해당하는 경우)를 사용해서 OBT 애플리케이션을 구성한다.
2) OBT 는 ECDSA 를 사용해서 Device 를 인증하여 서명을 확인해야 한다. 추가적으로, Device 는 OBT 서명을 확인하기 위해 OBT 를 인증하는 경우가 있다.
3) 인증을 실패하면, Device 는 실패 이유를 나타내고 Ready for OTM 상태로 되돌아가야 한다. 인증을 성공하면, device 와 OBT 는 협상된 cipher suite 에 따라 암호화 링크를 확립해야 한다.

7.3.6.1 인증서 프로파일

Device PKI 내에서, 인증서 내의 subject 에 대해 다음과 같은 형식이 사용되어야 한다. 확장성과 오류 해결을 위한 목적으로 N 개의 독립된 root 가 있는 것으로 가정한다. 공급자 생성 및 동작 root 는 인증서 정책 (CP) 문서 및 적합한 RFP 문서에 기술된 OCF 기반 프로세스에 의해 승인된다. 각 root 는 하나 이상의 DEV CA 를 발행하여 개별 제조사에게 차례로 제조사 DEV CA 를 발행한다. 제조사는 복수의 제조사 CA 를 요청할지를 결정할 수 있다. 각 제조사 CA 는 하나 이상의 Device Sub-CA (M 개까지)를 발행하고 하나 이상의 OCSP responder (O 개까지)를 발행한다. 현재, CA 인증서에 대한 폐기 확인은 상위 계층의 CA 에 의해 발행된 CRL 에 의해 다뤄지는 것으로 가정할 수 있다.



도 16 – 제조사 인증서 계층

- Root CA: C=<country where the root was created>, O=<name of root CA vendor>, OU=OCF Root CA, CN=OCF (R) Device Root-CA<n>
- DEV CA: C=<country for the DEV CA>, O=<name of root CA vendor>, OU=OCF DEV CA, CN=<name of DEV CA defined by root CA vendor>
- Manufacturer DEV CA: C=<country where Manufacturer DEV CA is registered>, O=<name of root CA vendor>, OU=OCF Manufacturer DEV CA, CN=<name defined by manufacturer><m>
- Device Sub-CA: C=<country device sub-CA>, O=<name of root CA vendor>, OU=OCF Manufacturer Device sub-CA, OU=<defined by Manufacturer>, CN=<defined by manufacturer>
- For Device Sub-CA Level OCSP Responder: C=<country of device Sub-CA>, O=<name of root CA vendor>, OU=OCF Manufacturer OCSP Responder <o>, CN=<name defined by CA vendor >
- Device cert: C=<country>, O=<manufacturer>, OU=Device, CN=<device Type><single space (i.e., " ")><device model name>
    - OU=OCF (R) Device 와 CN= naming element 간에 다음과 같은 옵션 명명 요소가 포함될 수 있다. 이들은 임의의 순서로 나타날 수 있다: OU=chipsetID: <chipsetID>,

OU=<device type>, OU=<device model name> OU=<mac address>
OU=<device security profile>

- Gateway Sub-CA: C=<country>, O=<manufacturer>, OU=<manufacture name> Gateway sub-CA, CN=<name defined by manufacturer>, <unique Gateway identifier generated with UAID method>
- Home Device Cert: C=<country>, O=<manufacturer>, OU=Non-Device cert, OU=<Gateway UAID>, CN=<device Typle>

Gateway Sub-CA 에 관한 기술 참조: 제조사가 자신의 Gateway 를 Gateway Sub-CA 로 기능하도록 하고자 할 때는 Device Sub-CA 인증서 내에서 경로 길이 제약 값에 적절한 값을 설정함으로써 이를 수용해서 후자의 sub-CA 가 Gateway Sub-CA 에 CA 인증서를 발행하도록 해야 한다. Gateway Sub-CA 의 수는 상당히 많을 수 있으므로 Gateway Sub-CA ID 에 대해 번호 할당 기법을 사용해야 하며, Gateway 가 공개 키 쌍을 갖고 있으므로 게이트웨이 공개 키의 해시를 사용하여 게이트웨이 식별자를 계산하기 위한 UAID 알고리즘이 사용되어야 하고, 이는 Gateway Sub-CA 인증서의 주체 필드 내에 삽입된다.

별도의 Device Sub-CA 가 Gateway Sub-CA 인증서를 생성하는데 사용되어야 한다. 이러한 Device Sub-CA 는 비 Gateway device 인증서를 발행하는데 사용하지 않아야 한다.

Gateway Sub-CA 인증서를 포함하는 CRL 은 Gateway Sub-CA 절충에 관련된 잠재적인 많은 책임을 피하기 위해 분기별보다는 월별로 발행해야 한다.

Gateway Sub-CA 에 의해 발행된 Device 인증서는 OCF 주관 CA 에 의해 발행되지 않았음을 가리키기 위해 OU=Non-Device cert 를 포함해야 한다.

명명 요소가 DirectoryString (i.e., O=, OU=)일 때는 PrintableString 또는 UTF8String 이 사용되어야 한다. 어떠한 선택을 사용할지는 다음과 같이 결정한다.

- PrintableString: 다음과 같은 US ASCII 문자 (ASN.1 에 의해 요구되는)의 부분집합에 제한될 때만:
  A, B, …, Z
  a, b, …, z
  0, 1, …9,
  (공백) ' ( ) + , - . / : = ?
- 그 밖의 모든 경우 UTF8String. 예를 들어, 그 어떤 다른 문자 또는 국제 문자 집합을 사용한 주체 명칭 속성.

CVC CA 는 신뢰성 있는 기구에 의해 소프트웨어 제공자, 시스템 관리자, 또는 그 밖의 Device 에 대한 소프트웨어 이미지에 서명하는 개체에 대해 인증서에 서명하기 위한 CVC 코드를 발행하는데

사용된다. CVC CA 는 코드 서명 이외에는 어떤 항목에 대해서도 인증서를 서명 및 발행하지 않아야 한다. 다시 말하면, CVC CA 는 CVC 분야 이외의 그 어떤 분야에 속하는 인증서에 대해서도 서명 및 발행하지 않아야 한다.

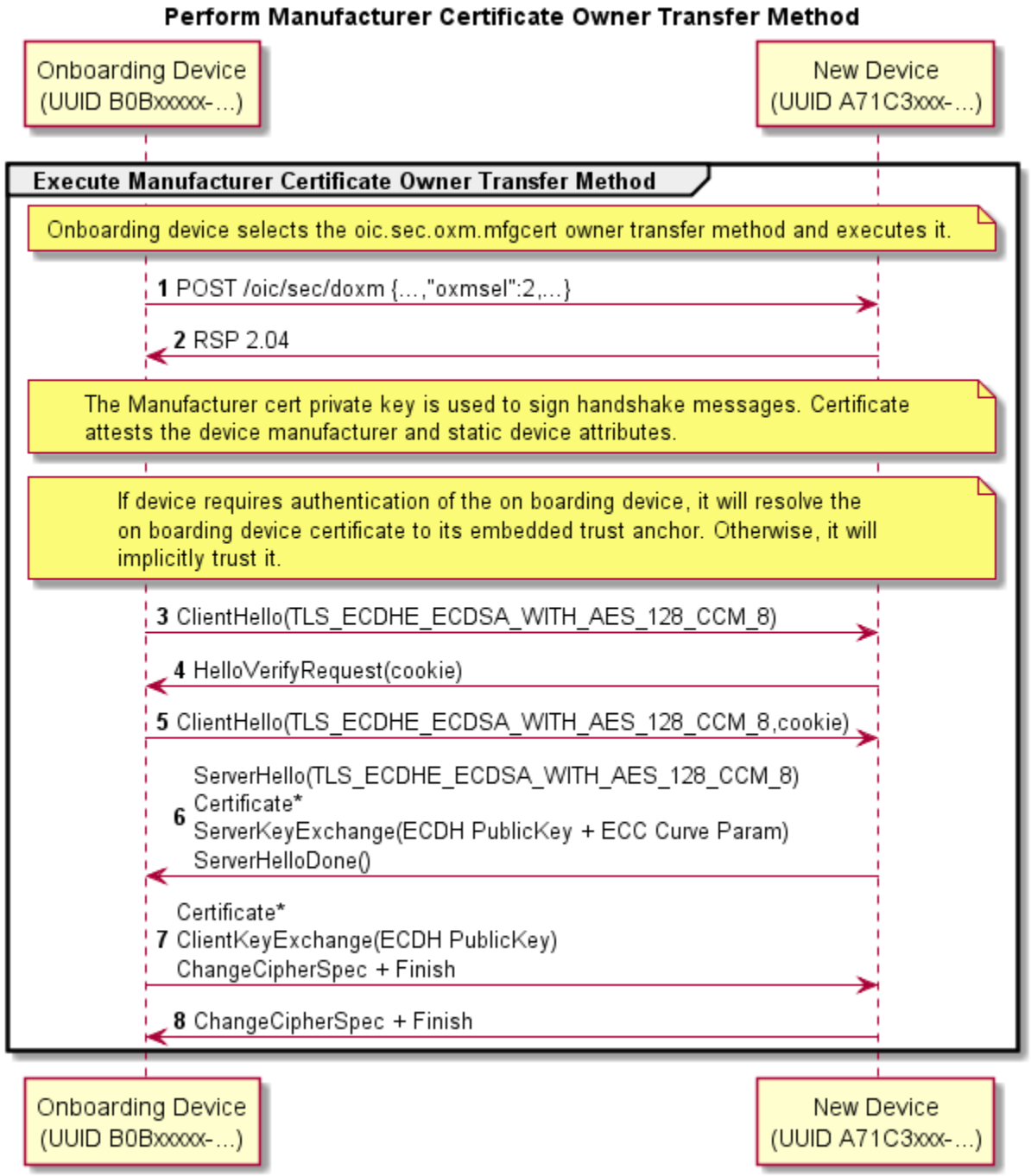
7.3.6.2 인증서 소유권 이전 시퀀스 보안 고려 사항

device 와 OBT 간의 완전한 상호 인증을 위해, device 와 OBT 는 둘 다 상호 신뢰 앵커 또는 인증 기관까지 거슬러 올라갈 수 있어야 한다. 이것은 OCF 가 이후의 모든 OCF Trust Anchor 가 도출되는 궁극적인 Trust Anchor 를 제공하기 위해 인증 기관 (예: Symantec, Verisign 등)으로부터 서비스를 취득할 필요가 있음을 암시한다.

OBT 는 온보딩 동안에 device 를 인증해야 한다. 그러나, device 는 device 상의 잠재적인 resource 제약으로 인해 OBT 를 인증하지 않아도 된다.

Device 가 OBT 소프트웨어를 인증하지 않는 경우, 사용자가 자신도 모르는 사이에 악의적인 OBT 소프트웨어를 배포하거나 적대적인 상대가 악의적으로 배포할 수 있으므로 이로 인해 네트워크 액세스 크리덴셜 및/또는 개인정보가 위태롭게 될 가능성이 있다.

1572 ### 7.3.6.3 제조사 인증서 기반 소유권 이전 방법 시퀀스



1573
1574
1575 **도 17 – 제조사 인증서 기반 소유권 이전 시퀀스**

단계	설명
1, 2	OBT 가 Device 에 '제조사 인증서' 방법을 선택하였음을 통지한다.
3 - 8	device 의 제조사 인증서 및 옵션 OBT 인증서를 사용해서 DTLS 세션이 확립된다. device 의 제조사 인증서는 Device 강화 및 보안 property 를 증명하는 데이터를 포함할 수 있다.

표 5 – 제조사 인증서 기반 소유권 이전 방법 상세

7.3.6.4 보안 고려 사항

제조사 인증서 개인 키는 개인 키가 침해되지 않을 정도의 충분한 보증을 갖고 Platform 에 내장된다.

Platform 제조사는 제조사 인증서를 발행하고 개인 키 보호 메커니즘을 증명한다.

제조사 인증서는 고유성 property 를 정의한다.

단일 제조사 인증서를 포함하는 Platform 에 의해 복수의 Device instance 가 호스트될 수 있다.

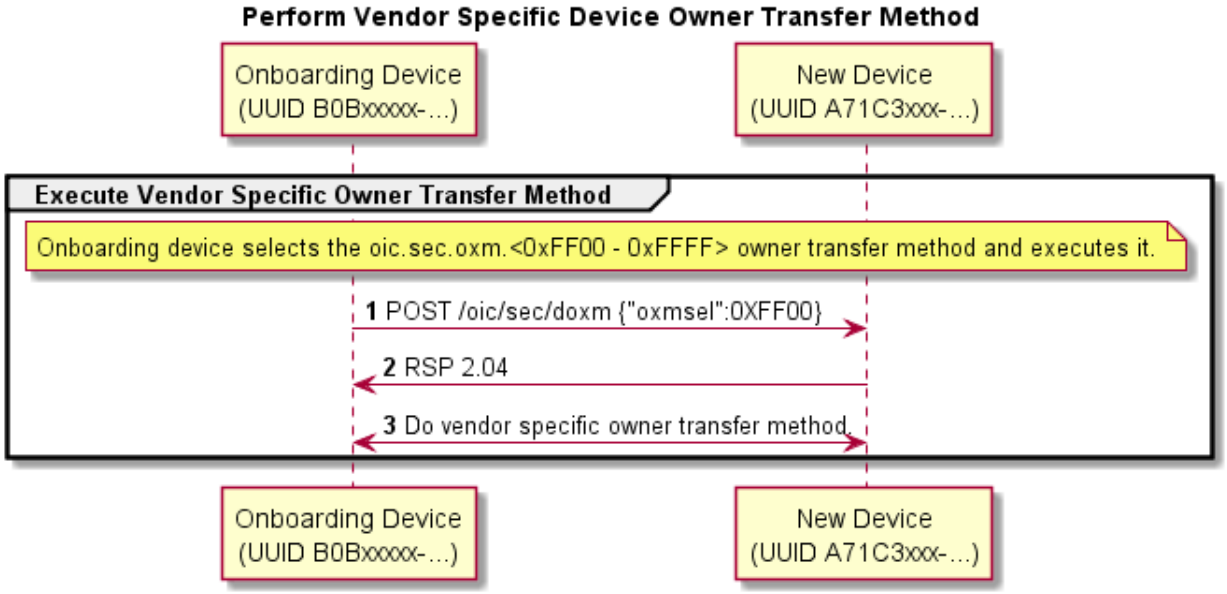
7.3.9 공급자 특정 소유권 이전 방법

OCF 는 공급자가 제조 또는 Device 제약을 수용하는 소유권 이전 방법을 구현해야 하는 상황을 예상한다. Device 소유권 이전 방법 resource 는 이러한 목적을 위해 확장 가능하다. 공급자 특정 소유권 이전 방법은 모든 소유권 이전 방법이 따르는 일련의 규약을 준수해야 한다.

- OBT 는 어떤 크리덴셜 타입이 Device 에 의해 지원되는지 결정해야 한다. 이것은 Device 의 /oic/sec/doxm Resource 에 문의해서 지원되는 크리덴셜 타입을 파악함으로써 이루어진다.

- OBT 는 OC 와 함께 Device 를 프로비저닝한다.
- OBT 는 OBT 에의 이후 액세스를 위한 Device ID 와 크리덴셜을 제공한다.
- OBT 소유권 설정 다음에 소유자의 네트워크에 액세스하기에 충분한 이차 캐리어 설정을 제공한다.
- OBT 는 추가적인 프로비저닝 단계를 수행할 수 있지만 부트스트랩 또는 보안 서비스에 의해 수행되는 프로비저닝 작업을 무효화해서는 안된다.

7.3.7.1 공급자 특정 소유권 이전 시퀀스 예



도 18 – 공급자 특정 소유권 이전 시퀀스

단계	설명
1, 2	OBT 가 공급자 특정 소유권 이전 방법을 선택한다.
3	공급자 특정 소유권 이전 방법이 적용된다.

표 6 – 공급자 특정 소유권 이전 상세

7.3.7.2 보안 고려 사항

공급자는 보안 위협 및 완화 전략을 고려할 책임이 있다.

7.3.8 소유자 크리덴셜 설정

OBT 와 신규 Device 의 상호 인증이 완료되면 정의된 OTM 방법 중 하나를 사용해서 암호화 연결이 확립된다.

소유자 크리덴셜은 OBT 또는 그 밖의 권위있는 개체에 의해 서명된 인증서, 사용자 네트워크 액세스 정보, 프로비저닝 기능, 공유 키, 또는 Kerberos 티켓으로 구성될 수 있다.

그리고 나서, OBT 는 Device 관리 및 Device 대 Device 통신을 위한 추가적인 크리덴셜을 신규 Device 에 프로비저닝할 수 있다. 이러한 크리덴셜은 서명, Device 공개 키, PSK 를 토대로 한 UAID 등을 갖는 인증서로 구성될 수 있다.

Device 의 소유자 크리덴셜 (OC)를 설정하기 위한 상세 단계는 다음과 같다.

a. OBT 가 Device ID 와 Device 소유자 uuid 를 설정해야 한다 - 도 19

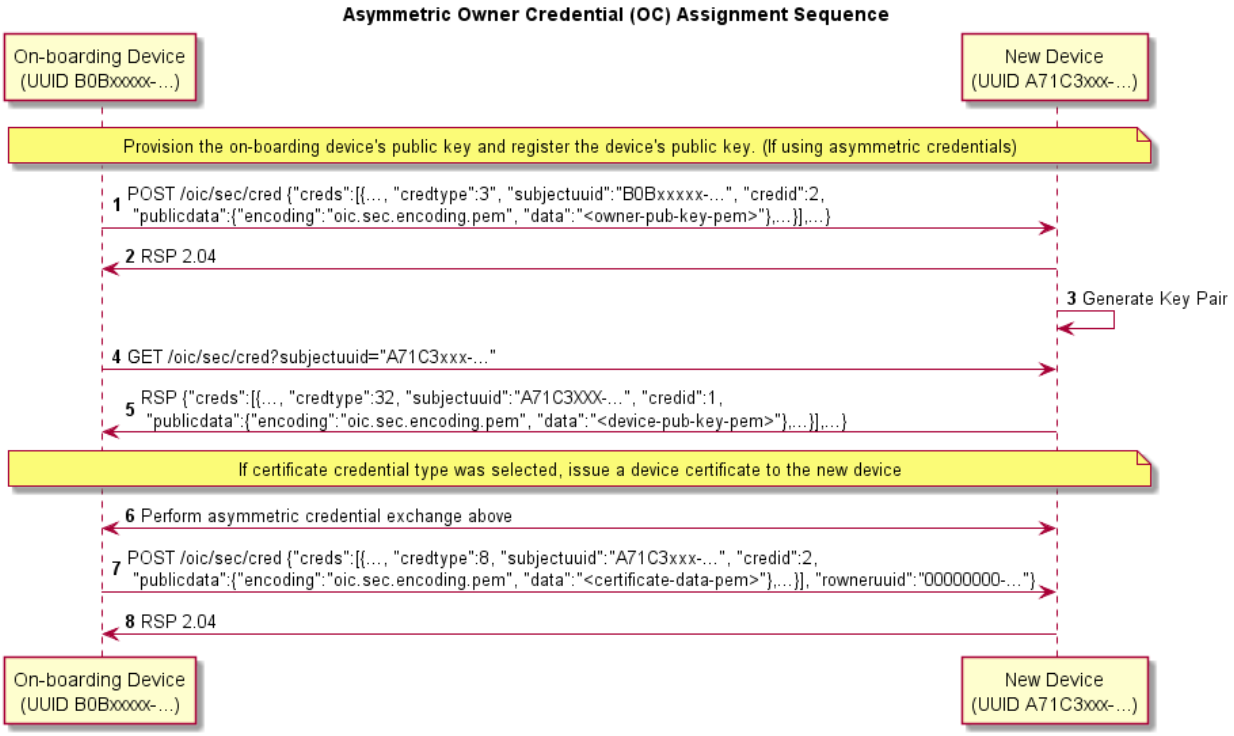
b. 그리고 나서, OBT 가 Device 의 소유자 크리덴셜 (OC)를 설정한다 - 도 20. 이것은 다음 중 하나일 수 있다.

i. 대칭 크리덴셜 - 도 21

@startuml

ii. autonumber

iii. title Asymmetric Owner Credential (OC) Assignment Sequence



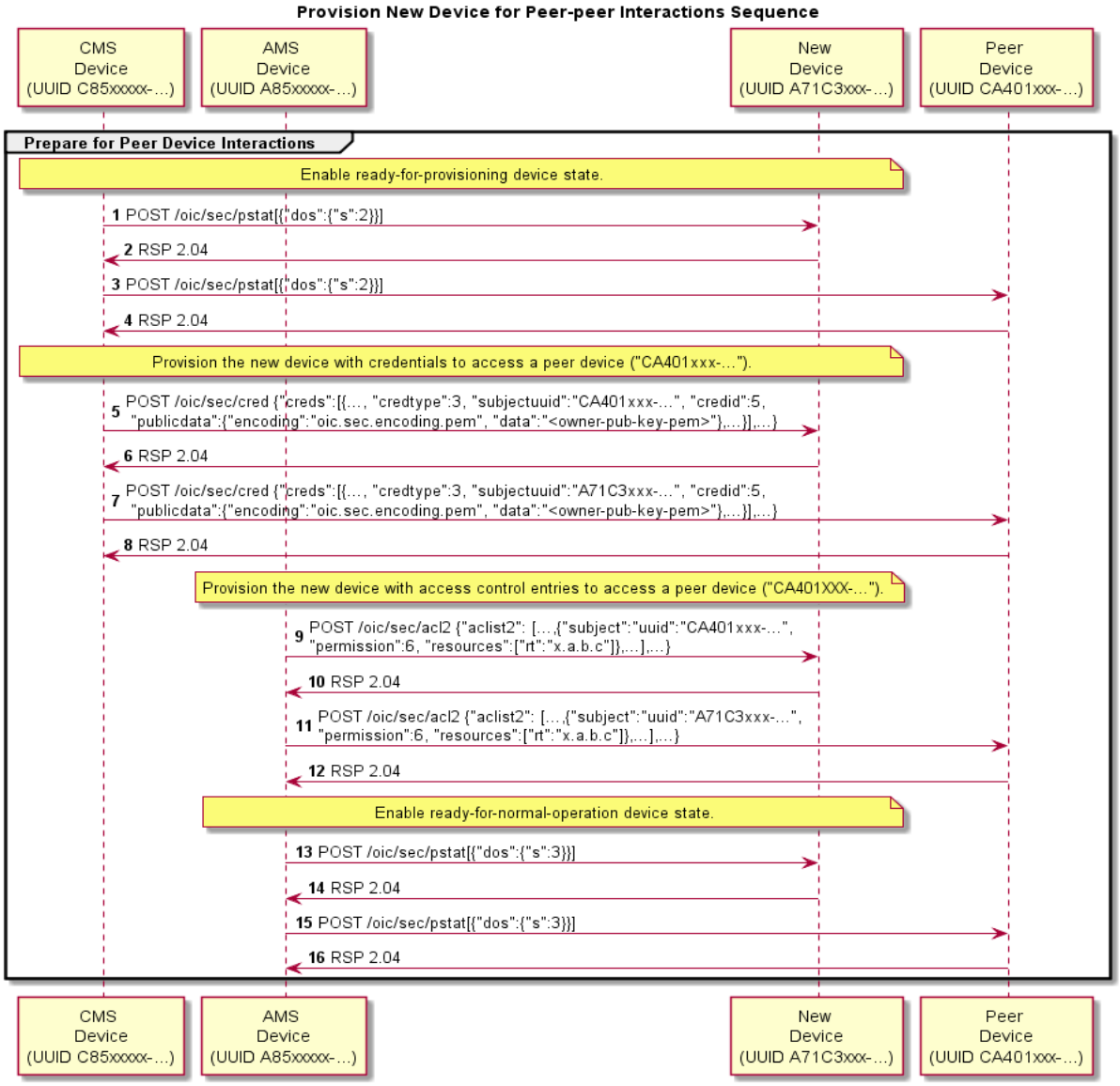
iv. 도 22

c. Device 서비스 구성. - 도 23

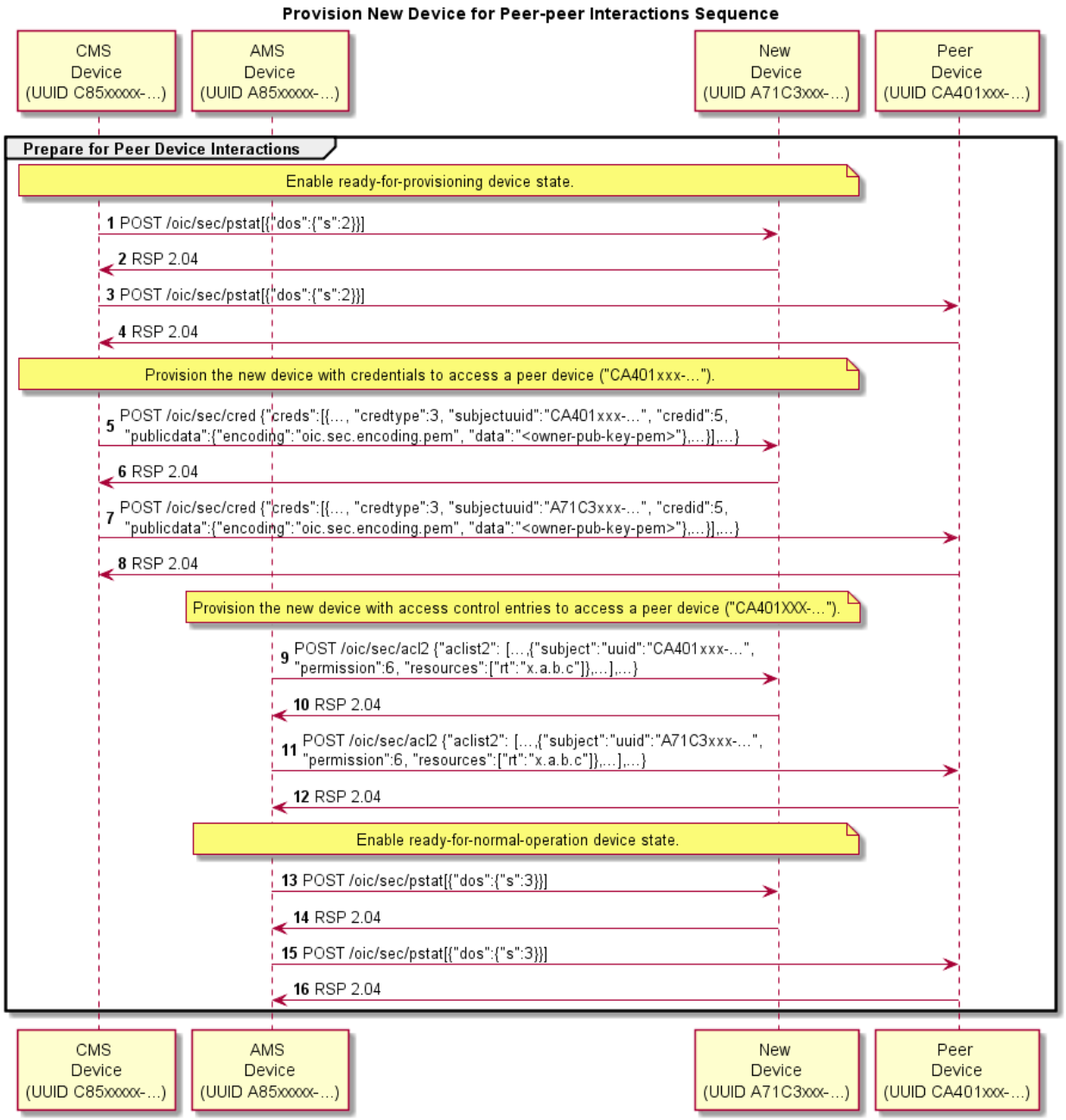
@startuml

d. autonumber

e. title Provision New Device for Peer-peer Interactions Sequence

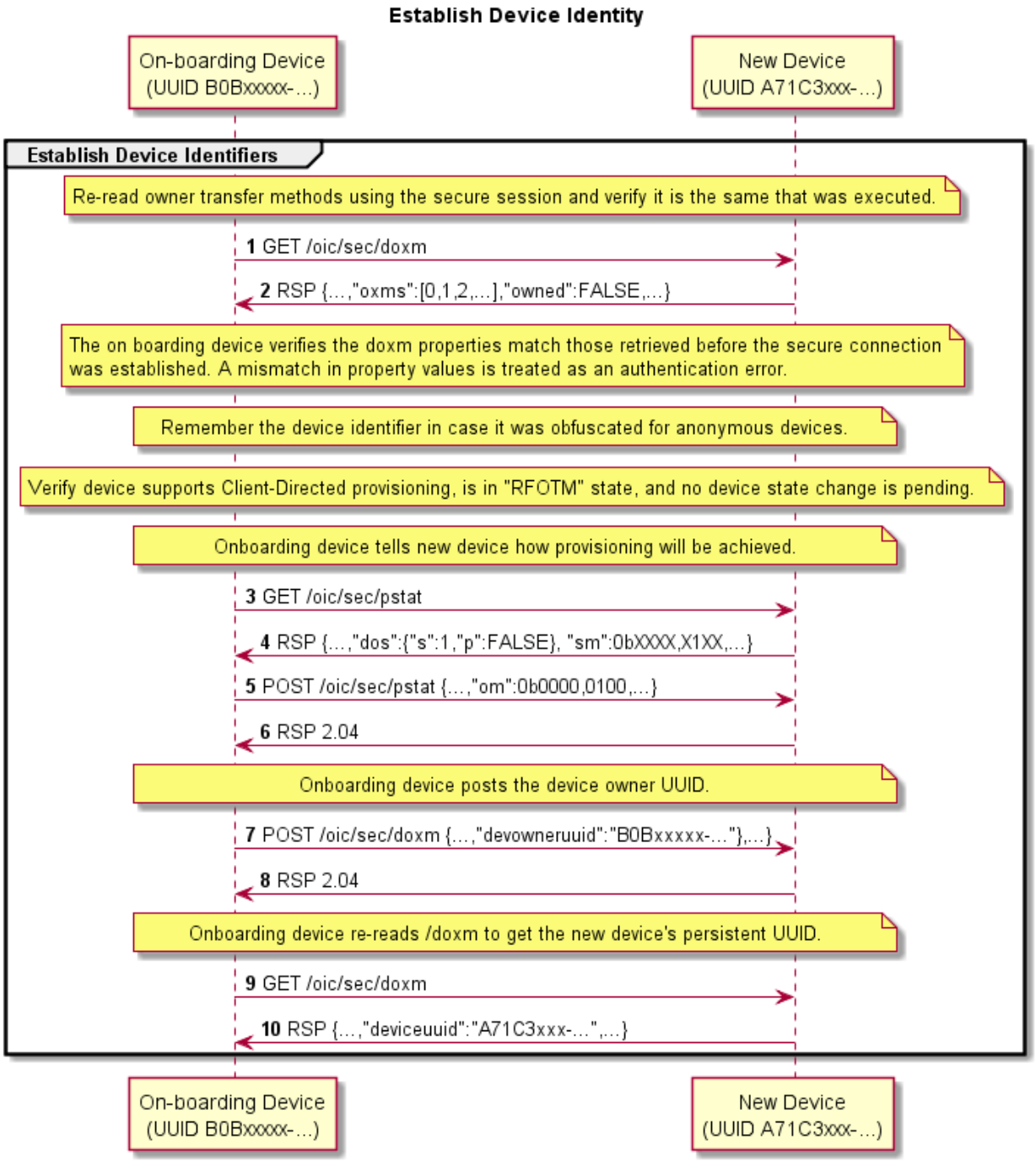


f. 도 24



이들 크리덴셜은 OBT 또는 그 밖의 권위있는 개체에 의해 서명된 인증서, 사용자 네트워크 액세스 정보, 프로비저닝 기능, 공유 키, 또는 Kerberos 티켓으로 구성될 수 있다.

그리고 나서, OBT 는 Device 관리 및 Device 대 Device 통신을 위한 추가적인 크리덴셜을 신규 Device 에 프로비저닝할 수 있다. 이러한 크리덴셜은 서명, Device 공개 키, PSK 를 토대로 한 UAID 등을 갖는 인증서로 구성될 수 있다.

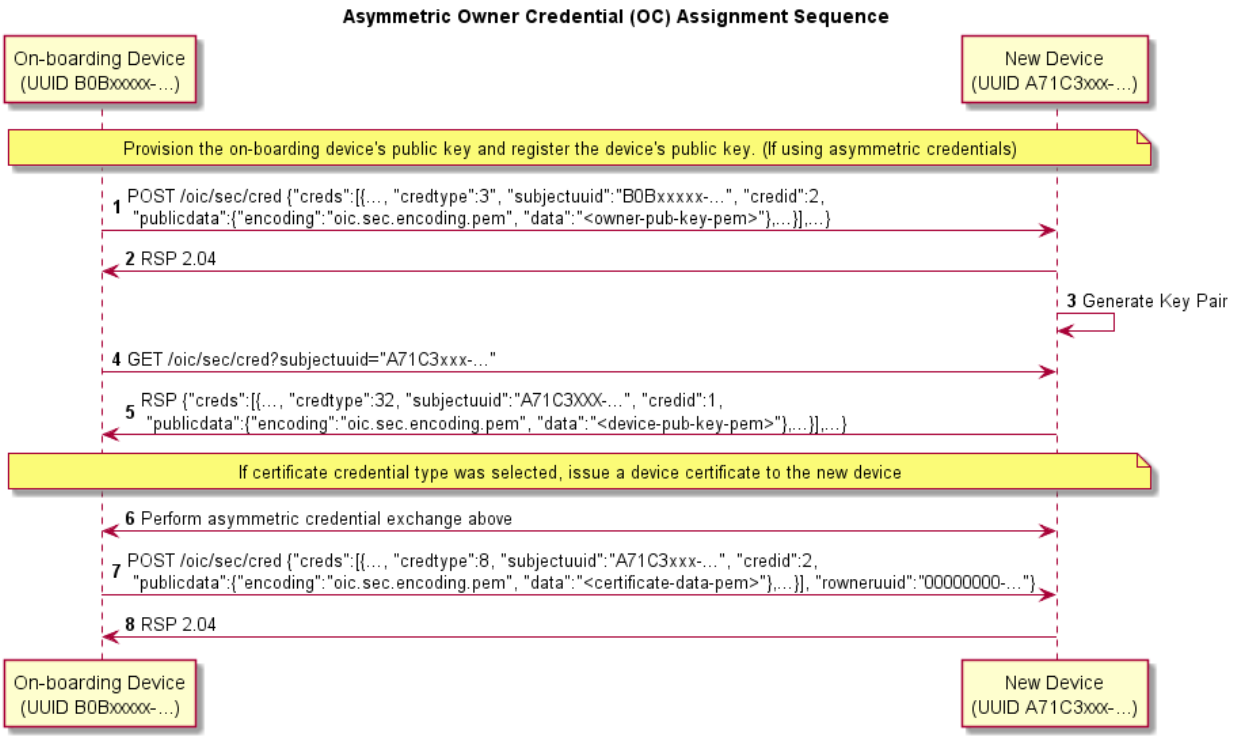


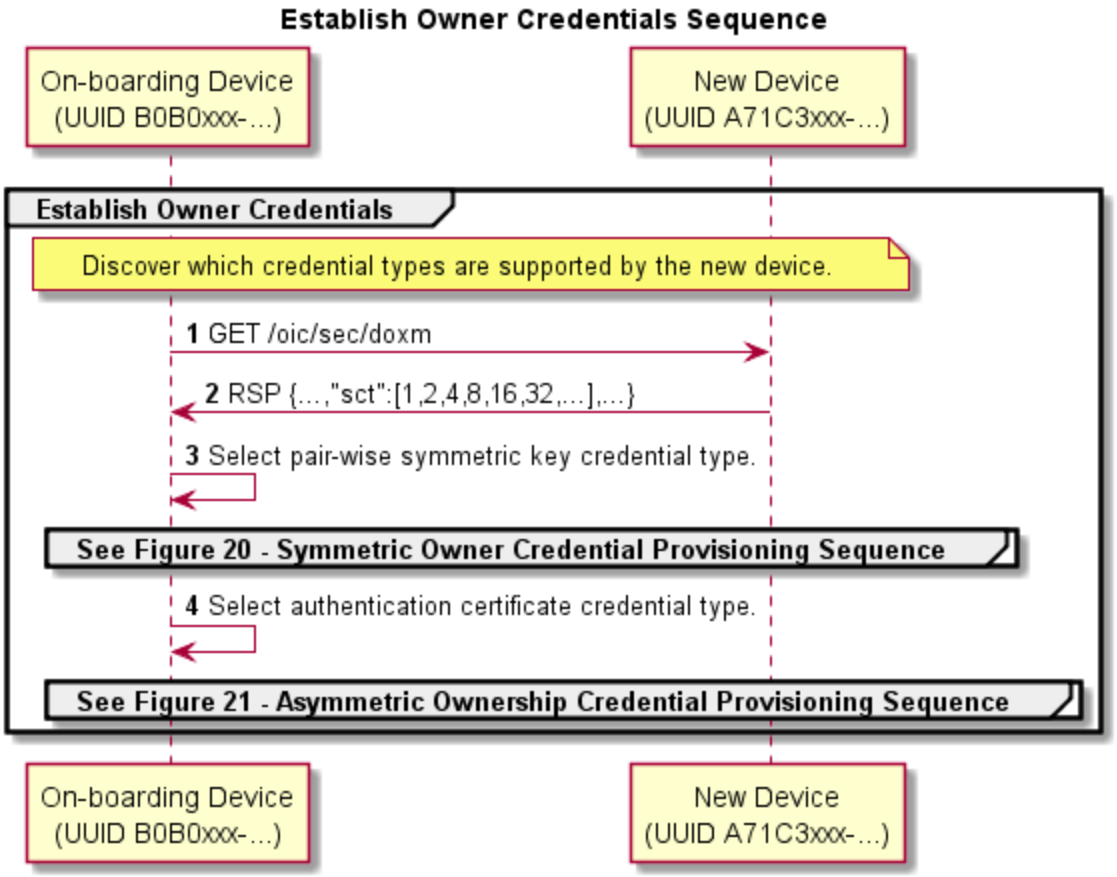
도 19 - Device 아이덴티티 설정 흐름

단계	설명
1, 2	OBT 가 보안 세션을 사용해서 doxm property 를 다시 취득한다. 이들 property 가 인증된 연결 전에 검색된 것과 일치하는 것을 확인한다. 파라미터의 불일치는 인증 에러로 취급된다.
3, 4	OBT 가 질의를 통해 Device 가 Device 소유권 이전 준비가 완료되어 있는지 파악한다.
5, 6	OBT 가 Client 프로비저닝 규약을 따를 것을 주장한다.
7, 8	OBT 가 Device ID 를 자신의 ID 로 설정해서 자신이 신규 Device 의 소유자임을 주장한다.
9, 10	OBT 가 doxm property 를 다시 취득한다. 이 때 Device 는 신규 Device 의 지속적인 UUID 를 리턴한다.

표 7 - Device 아이덴티티 설정 상세

group See @startuml



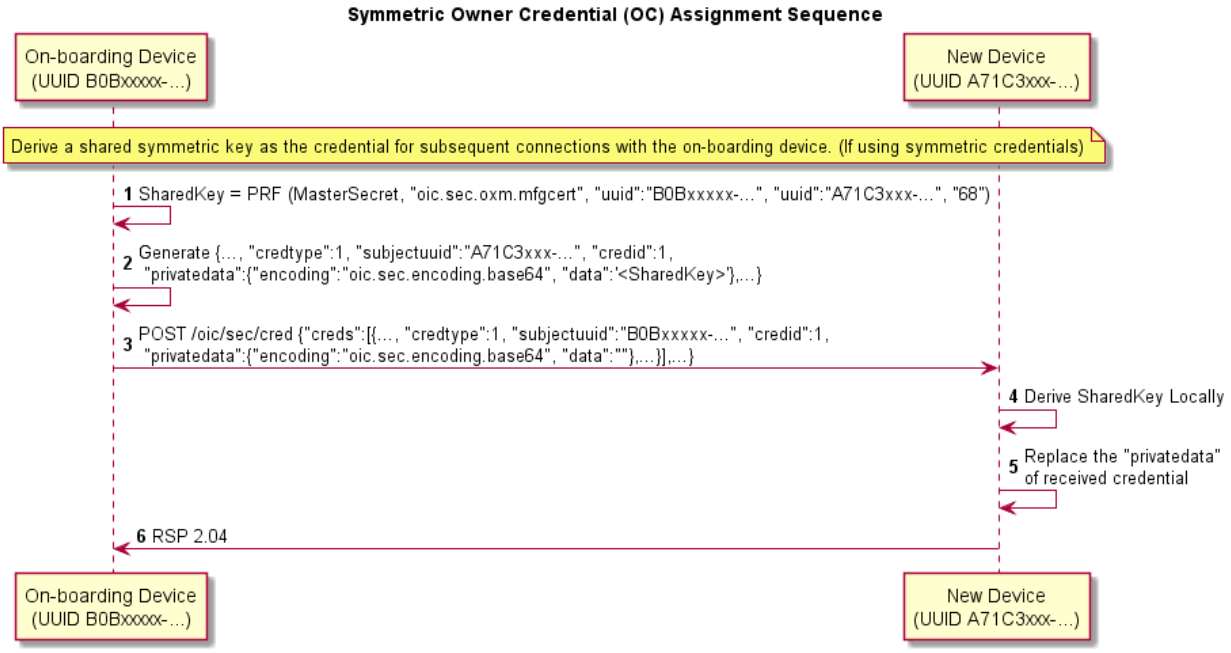


1643

1644 **도 20 – 소유자 크리덴셜 선택 프로비저닝 시퀀스**

단계	설명
1, 2	OBT 가 doxm property 를 취득해서 신규 Device 에 의해 지원되는 소유권 이전 메커니즘을 확인한다.
3, 4	OBT 가 소유권 프로비저닝을 위해 선택한 크리덴셜 타입을 사용한다.

1645 **표 8 – 소유자 크리덴셜 할당 상세**



1646

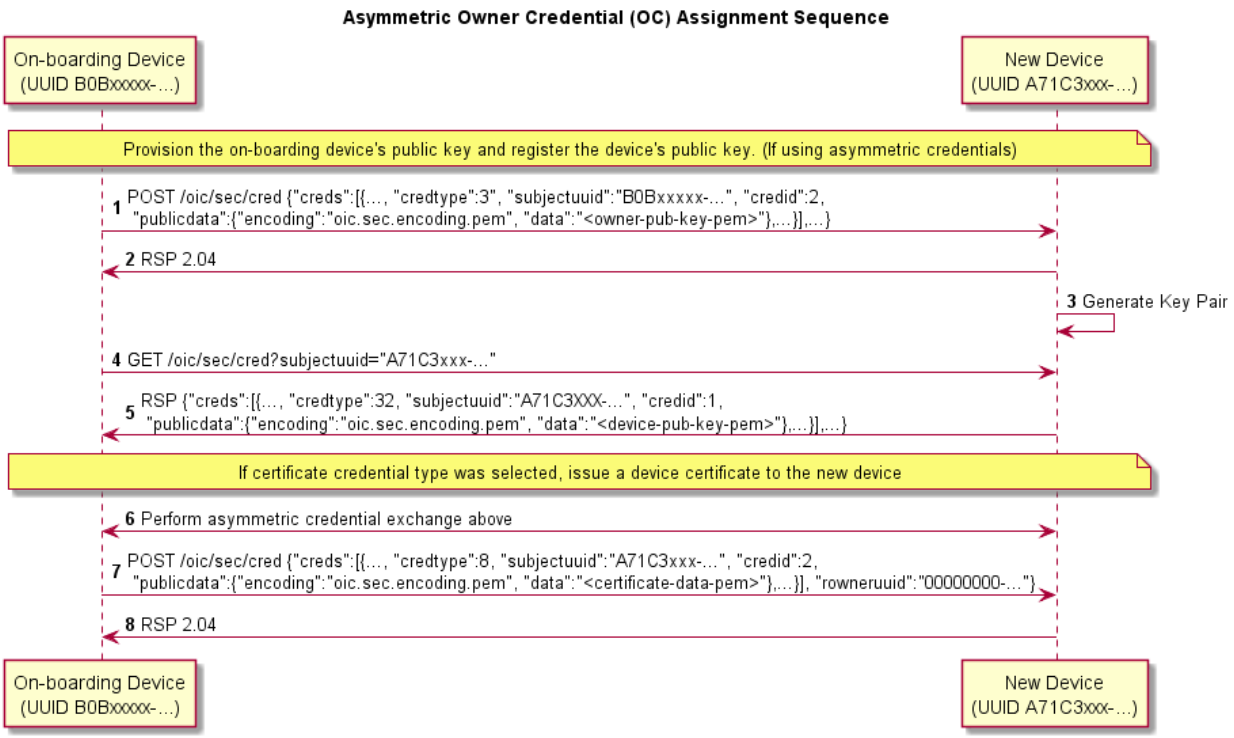
1647 **도 21 - 대칭 소유자 크리덴셜 프로비저닝 시퀀스**

단계	설명
1, 2	OBT 가 pseudo-random-function (PRF), DTLS 핸드세이크로부터 기인하는 마스터 비밀, 및 그 밖의 정보를 사용해서 대칭 키 크리덴셜 resource property - SharedKey 를 생성한다.
3	OBT 가 SharedKey 를 토대로 크리덴셜 resource property 집합을 생성해서 빈 "privatedata" property 값과 함께 resource property 집합을 신규 Device 로 전송한다.
4, 5	신규 Device 가 로컬에서 SharedKey 를 생성하고 credential resource property 집합의 "privatedata" property 로 갱신한다.
6	신규 Device 가 성공 메시지를 전송한다.

표 9 - 대칭 소유자 크리덴셜 이전 상세

특히, OBT 가 대칭 소유자 크리덴셜을 선택하면,

- OBT 는 섹션 **Error! Reference source not found.**에 기술된 SharedKey Credential Calculation 방법을 사용해서 Shared Key 를 생성해야 한다.
- OBT 는 대칭 신규 Device 의 /oic/sec/cred Resource 로 쌍 키로 식별되는 빈 키를 전송해야 한다.
- OBT 의 대칭 소유자 크리덴셜을 수신하면, 신규 Device 는 섹션 **Error! Reference source not found.**에 기술된 SharedKey Credential Calculation 방법을 사용해서 개별적으로 Shared Key 를 생성하고 이를 소유자 크리덴셜과 함께 저장해야 한다.
- 신규 Device 는 /oic/sec/cred Resource 를 통해 저장된 Shared Key 소유자 크리덴셜을 사용해서 이후의 연결 시에 소유자를 인증해야 한다.



도 22 - 비 대칭 소유권 크리덴셜 프로비저닝 시퀀스

단계	설명
OBT 에 의해 비 대칭 또는 인증서 소유자 크리덴셜 타입이 선택되었을 때	
1, 2	OBT 가 신규 Device 에 공개 키 (OC)와 함께 비 대칭 타입 크리덴셜 Resource property 집합을 생성해서 전송한다. 이것은 이후에 OBT 를 인증하는데 사용될 수 있다. 신규 device 는 공개 키를 토대로 크리덴셜 Resource property 를 생성한다.
3	신규 Device 가 비 대칭 키 쌍을 생성한다.
4, 5	OBT 가 2 단계에서 생성한 신규 Device 의 비 대칭 타입 크리덴셜 Resource property 집합을 읽는다. 이것은 이후에 신규 Device 를 인증하는데 사용될 수 있다.
OBT 에 의해 인증서 소유자 크리덴셜 타입이 선택되었을 때	
6-8	비 대칭 크리덴셜 타입을 생성하는 단계가 수행된다. 또한, OBT 가 신규 Device 에 대해 새로 생성된 인증서 (또는 인증서 체인)를 인스턴스화한다.

표 10 – 비 대칭 소유권 크리덴셜 할당 상세

OBT 가 비 대칭 소유자 크리덴셜을 선택하면,

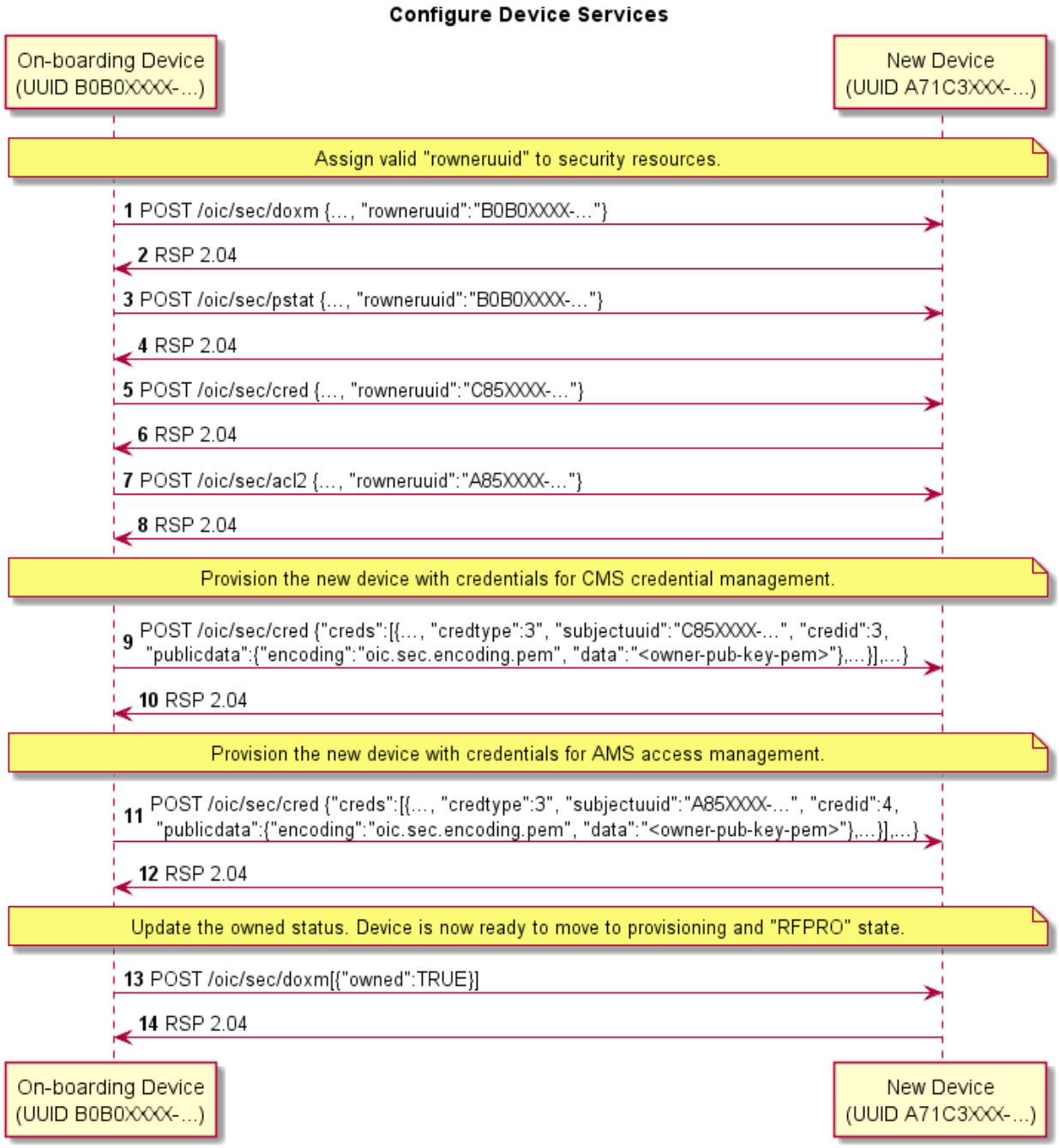
- OBT 는 신규 Device 의 /oic/sec/cred Resource 에 비 대칭 암호 키로서 식별되는 자신의 공개 키를 추가해야 한다.

- OBT 는 신규 Device 의 UUID 를 SubjectID query 파라미터로 제공하여 신규 Device 로부터 /oic/sec/cred Resource 를 질의해야 한다. 이에 대한 응답으로, 신규 Device 는 OBT 가 신규 Device 의 향후 소유자 인증을 위해 유지해야 하는 공개 비 대칭 암호 키를 리턴해야 한다.

OBT 가 인증서 소유자 크리덴셜을 선택하면,

- OBT 는 신규 Device 에 의해 리턴된 공개 키를 포함하고, 상호 신뢰하는 CA 에 의해 서명되고, 섹션 **Error! Reference source not found.**에 정의된 인증서 크리덴셜 생성 요구 사항을 준수하는 리프 인증서로 인증서 또는 인증서 체인을 생성해야 한다.

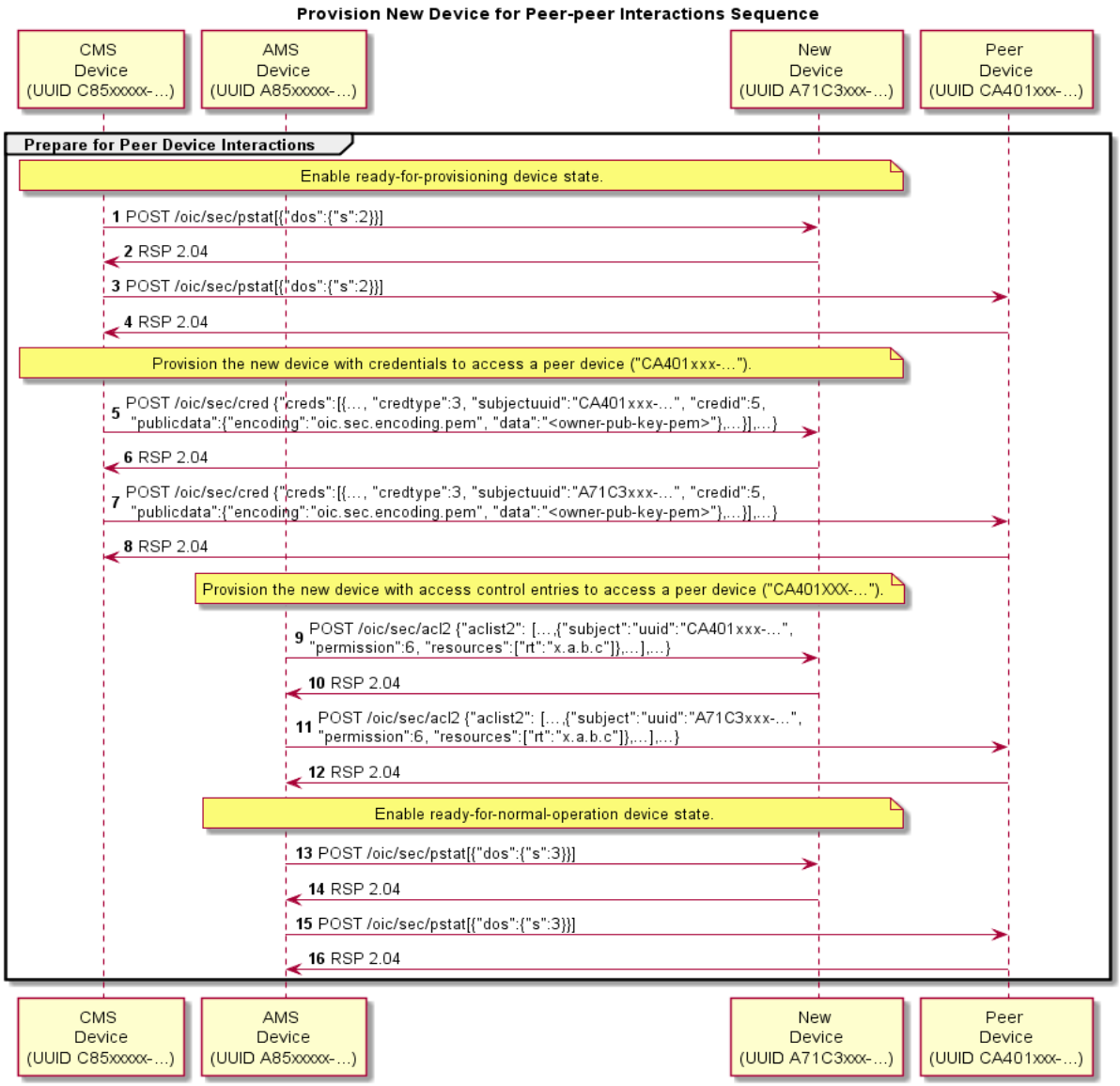
- OBT 는 /oic/sec/cred Resource 에 인증서를 가진 비 대칭 서명 키로서 식별되는 새로 생성된 인증서 체인을 추가해야 한다.



도 23 - Device Service 구성

단계	설명
1 - 8	OBT 가 SVR 에 대해 rowneruuid 를 할당한다.
9 - 10	신규 Device 에 CMS 용 크리덴셜을 프로비저닝한다.
11 - 12	신규 Device 에 AMS 용 크리덴셜을 프로비저닝한다.
13 - 14	oic.sec.doxm.owned 를 TRUE 로 갱신한다. Device 가 프로비저닝 및 RFPRO 상태로의 전환 대기 상태로 된다.

1679 **표 11 - Device Service 구성 상세**



1680
1681 **도 24 - P2P 연동을 위한 신규 Device 프로비저닝 시퀀스**

단계	설명
1 - 4	OBT 가 oic.sec.pstat.dos 를 2 로 설정함으로써 Device 를 프로비저닝 대기 상태로 설정한다.
5 - 8	OBT 가 Device 에 피어 크리덴셜을 프로비저닝한다.
9 - 12	OBT 가 Device 에 피어 Device 에 대한 액세스 제어 개체를 프로비저닝한다.
13 - 16	oic.sec.pstat.dos 를 3 으로 설정함으로써 Device 를 RFNOP 상태로 활성화한다.

표 12 - P2P 용 신규 Device 프로비저닝 상세

7.3.9 소유권 이전 방법 선택 시의 보안 고려 사항

OBT 및/또는 OBT 의 사용자에게는 신규 Device 의 소유권 이전 시에 받아들일 수 있는 OTM 목록에 대한 엄격한 요구 사항이 적용된다. 이러한 요구 사항을 결정할 때 고려되어야 하는 요인 중 일부는 다음과 같다.

- 각 OTM 에 대해 위에 기술한 보안 고려 사항.
- Ownership Transfer 를 수행하는데 사용되는 환경에 중간 공격자가 존재할 확률.

예를 들어, OBT 의 사용자는 온보딩되는 모든 Device 가 Random PIN 또는 제조사 인증서 OTM 을 지원할 것을 요구할 수 있다.

그러한 로컬 OTM 정책이 존재하면, OBT 는 위의 시퀀스로부터 1 단계에서 취득한 doxm 컨텐츠(GET /oic/sec/doxm)에 관계 없이 정책이 허용하는 OTM 을 사용하는 것이 좋다. 1 단계가 OBT 와 Device 간의 비 인증 및/또는 비 암호화 연결을 통해 수행되면 GET request 에 대한 response 의 내용이 중간 공격자에 의해 조작될 가능성이 있다. 예를 들어, 신규 Device 에 의해 지원되는 OTM 목록이 공격자에 의해 변조될 수 있다.

뿐만 아니라, 중간 공격자가 OBT 와 신규 Device 간의 DTLS 세션을 실패하도록 할 수 있다. 그러한 경우, OBT 는 신규 Device 가 OBT 에 의해 선택된 OTM 을 지원하지 않거나 중간 공격자가 OBT 와 신규 Device 간의 통신을 실패하도록 하였기 때문에 세션이 실패했는지 결정할 방법이 없다.

본 스펙의 현재 버전은 위에 언급한 OTM 정책에 관련된 설계 및 사용자 경험을 OBT 구현 상세로 남겨둔다.

7.4 프로비저닝

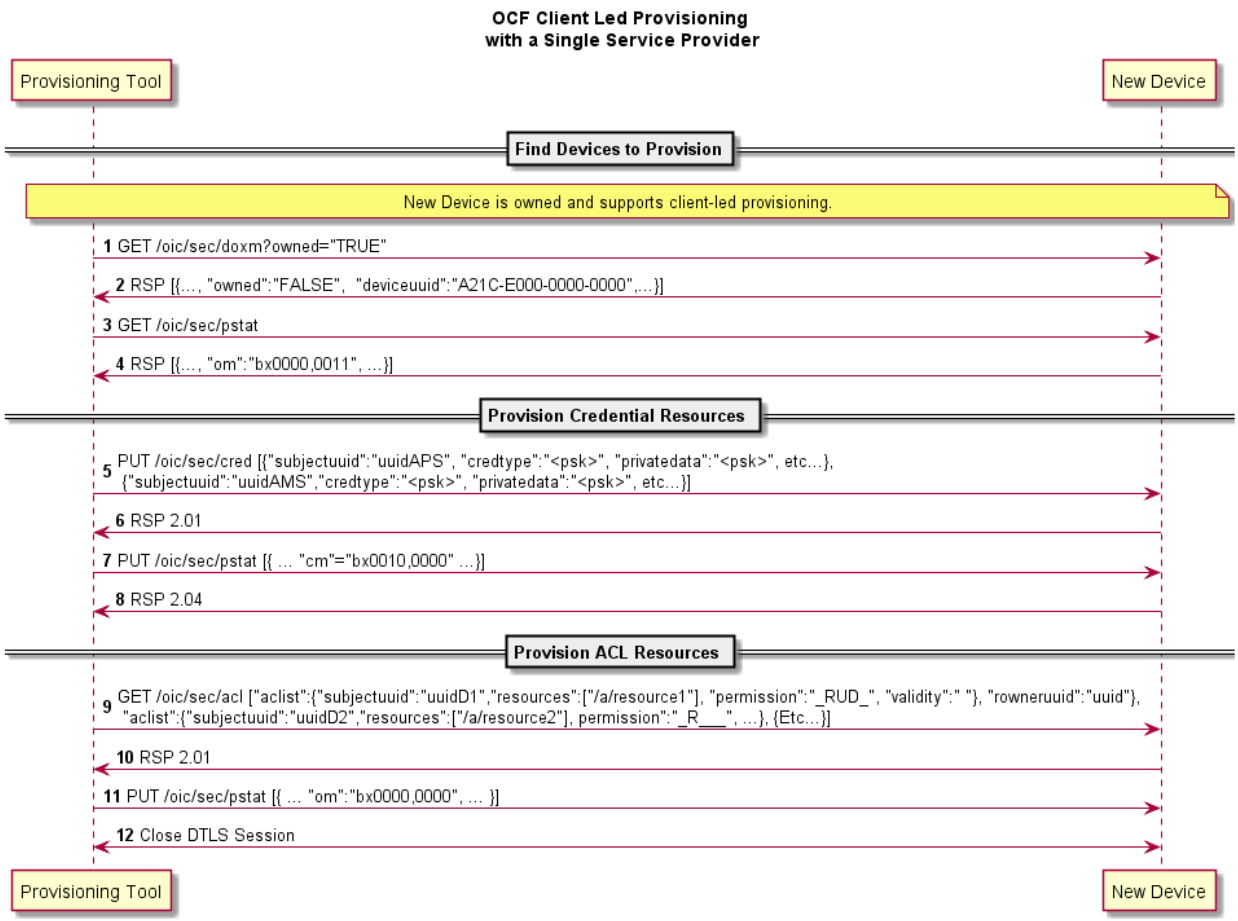
7.4.1 프로비저닝 절차

신규 Device 의 온보딩의 일환으로 신규 Device 와 OBT 간에 보안 채널이 형성된다. Device 의 소유권 상태가 '소유된'으로 변경된 후, 프로비저닝을 시작할 수 있다. OBT 는 앞으로 신규 Device 를 어떻게 관리할지를 결정하고 다음에 Device 프로비저닝 및 지속적인 Device 관리를 완료하는데 사용해야 하는 지원 서비스를 프로비저닝한다.

Device 는 Server 지향 또는 Client 지향 프로비저닝 전략을 사용한다. /oic/sec/pstat Resource 는 프로비저닝 전략 및 현재 프로비저닝 상태를 식별한다. 프로비저닝 서비스는 네트워크에 대해 가장 적합한 프로비저닝 전략을 결정해야 한다. 더 자세한 사항은 섹션 13.7 을 참조하기 바란다.

7.4.1.1 Client 지향 프로비저닝

Client 지향 프로비저닝은 프로비저닝을 필요로 하는 Server 를 식별하고 모든 필요한 프로비저닝 업무를 수행하는 프로비저닝 서비스에 의존한다.



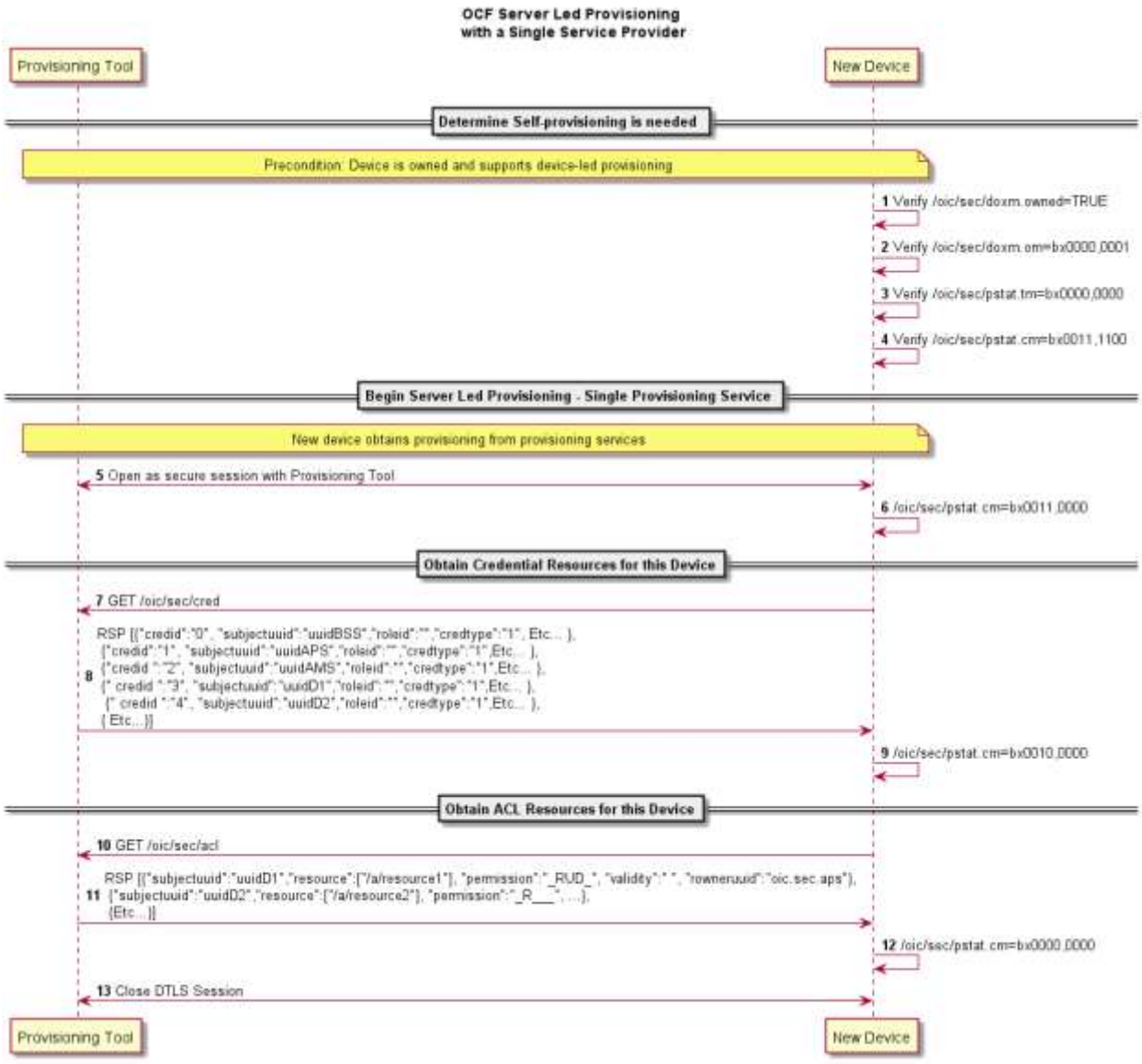
도 25 – Client 지향 프로비저닝 예

단계	설명
1	Client 지향 프로비저닝을 지원하는 소유된 상태의 Device 를 탐색한다.
2	/oic/sec/doxm Resource 가 Device 와 소유 상태를 식별한다.
3	PT 가 /oic/sec/pstat Resource 에서 발견된 신규 Device 의 프로비저닝 상태를 취득한다.
4	pstat Resource 가 현재 구성된 지원되는 프로비저닝 모드의 타입을 기술한다. Device 제조사는 초기 현재 작동 가능 모드 (om)를 설정해야 한다. Client 지향 프로비저닝에 대해 Om 이 구성되어 있지 않으면 om 값을 변경할 수 있다.
5 - 6	상태를 Ready-for-Provisioning 으로 변경한다. cm 은 프로비전 크리덴셜 및 ACL 로 설정된다.
7 - 8	PT 가 /oic/sec/cred Resource 를 인스턴스화한다. 이는 프로비저닝된 서비스와 그 밖의 Device 에 대한 크리덴셜을 포함한다.
9 - 10	cm 이 ACL 프로비저닝으로 설정된다.
11 - 12	PT 가 /oic/sec/acl Resource 를 인스턴스화한다.
13 -14	신규 Device 프로비저닝 상태 모드가 해당하는 ACL 이 구성되었음을 반영하도록 갱신된다. (Ready-for-Normal-Operation 상태)
15	보안 세션이 종료된다.

표 13 – Client 지향 프로비저닝 단계

7.4.1.2 Server 지향 프로비저닝

Server 지향 프로비저닝은 프로비저닝 작업의 대부분을 총괄하는 Server (즉, New Device)에 의존한다. 온보딩 프로세스의 일환으로 추가적인 프로비저닝을 추구하기 위해 Server 에 의해 사용되는 지원 서비스가 프로비저닝된다. New Device 는 자발적인 상태 기반 접근을 사용해서 현재 프로비저닝 상태를 분석하고 목표 상태로 진행한다. 이 예에서는 신규 Device 의 프로비저닝에 단일 지원 서비스가 사용되는 것으로 가정한다.



1728

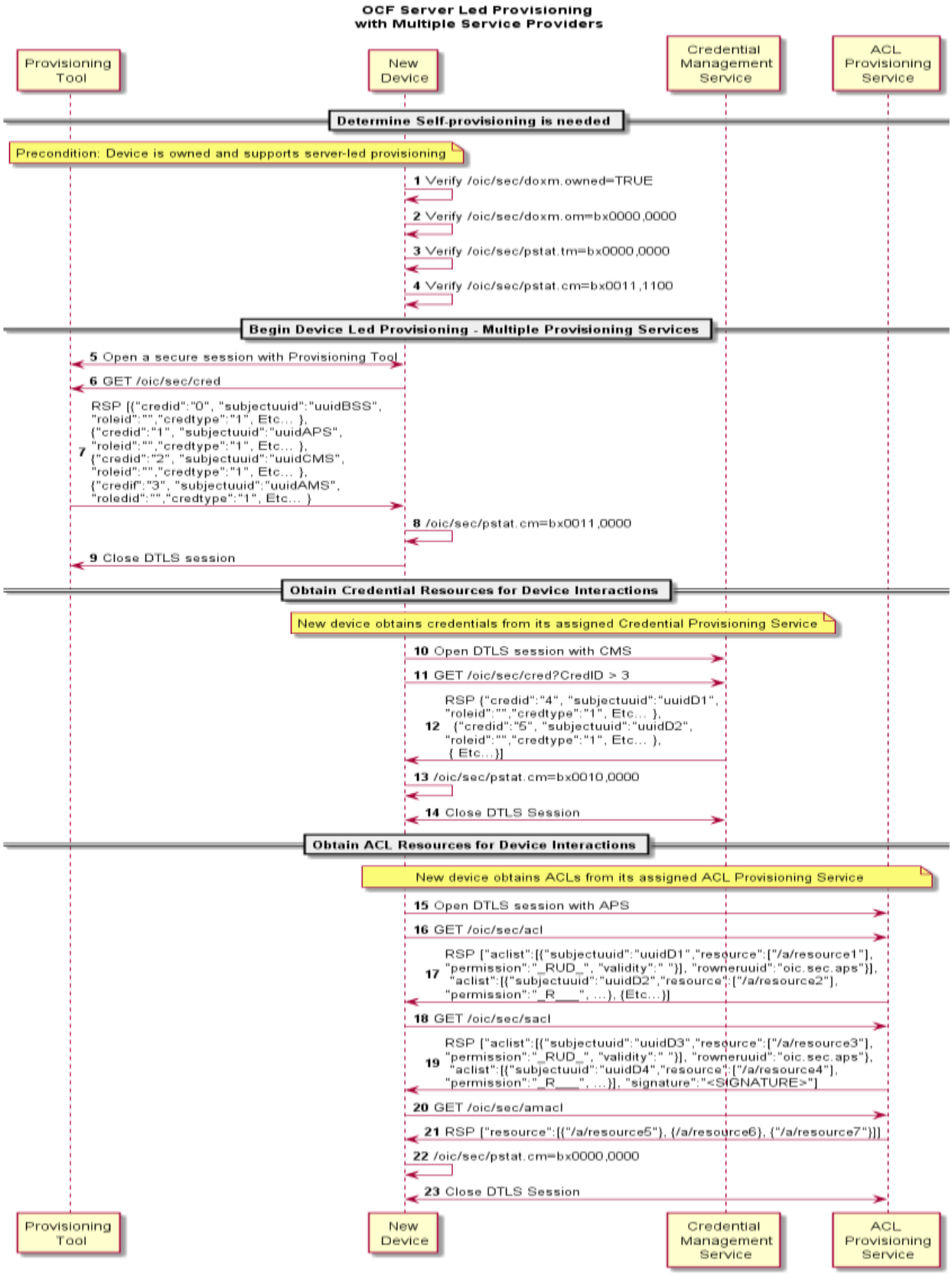
1729 **도 26 – 단일 프로비저닝 서비스를 사용한 Server 지향 프로비저닝 예**

단계	설명
1	신규 Device 가 소유된 상태임을 확인한다.
2	신규 Device 가 셀프 프로비저닝 모드로 되어 있는 것을 확인한다.
3	신규 Device 가 타겟 프로비저닝 상태가 전체 프로비저닝임을 확인한다.
4	신규 Device 가 자신의 현재 프로비저닝 상태가 프로비저닝을 필요로 하는 것을 확인한다.
5	신규 Device 가 /oic/sec/doxm 을 사용해서 프로비저닝 툴로 보안 세션을 개시한다. DevOwner 값은 SharedKey 를 사용한 TLS 연결을 개방하기 위한 것이다.
7	신규 Device 가 부트스트랩 및 그 밖의 서비스의 프로비저닝을 반영하도록 Cm 을 갱신한다.
8 – 9	신규 Device 가 /oic/sec/cred Resource 를 취득한다. 이것은 프로비저닝된 서비스와 그 밖의 Device 에 대한 크리덴셜을 포함한다.
10	신규 Device 가 크리덴셜 Resource 의 프로비저닝을 반영하도록 Cm 을 갱신한다.
11 – 12	신규 Device 가 /oic/sec/acl Resource 를 취득한다.
13	신규 Device 가 ACL Resource 의 프로비저닝을 반영하도록 Cm 을 갱신한다.
14	보안 세션이 종료된다.

표 14 – 단일 프로비저닝을 사용한 Server 지향 프로비저닝 단계

7.4.1.3 복수의 지원 서비스를 사용하는 Server 지향 프로비저닝

복수의 지원 서비스를 수반하는 Server 지향 프로비저닝의 흐름은 프로비저닝 작업을 복수의 지원 서비스에 분배한다. 복수의 지원 서비스의 사용은 프로비저닝 작업량을 분배하거나 전문화된 서비스를 전개하기 위한 효과적인 방법이다. 다음의 예는 프로비저닝 툴을 사용하여 크리덴셜 관리 지원 서비스와 ACL 프로비저닝 지원 서비스의 두 개의 지원 서비스를 구성하는 것을 보여준다.



도 27 – 복수의 지원 서비스를 포함하는 Server 지향 프로비저닝 예

단계	설명
1	신규 Device 가 소유된 상태임을 확인한다.
2	신규 Device 가 셀프 프로비저닝 모드로 되어 있는 것을 확인한다.
3	신규 Device 가 목표 프로비저닝 상태가 전체 프로비저닝임을 확인한다.
4	신규 Device 가 자신의 현재 프로비저닝 상태가 프로비저닝을 필요로 하는 것을 확인한다.
5	신규 Device 가 /oic/sec/doxm 을 사용해서 프로비저닝 툴로 보안 세션을 개시한다. DevOwner 값은 SharedKey 를 사용한 TLS 연결을 개방하기 위한 것이다.
6	신규 Device 가 지원 서비스의 프로비저닝을 반영하도록 Cm 을 갱신한다.
7	신규 Device 가 프로비저닝 툴로 DTLS 세션을 종료한다.
8	신규 Device /oic/sec/cred Resource 로부터 CMS 와 rowneruuid property 를 찾고 DTLS 연결을 개방한다. 신규 Device 가 /oic/sec/cred Resource 로부터 사용할 크리덴셜을 찾는다.
9 – 10	신규 Device 가 다른 device 와의 연동에 필요한 추가적인 크리덴셜을 요청한다.
11	신규 Device 가 크리덴셜 Resource 의 프로비저닝을 반영하도록 Cm 을 갱신한다.
12	DTLS 연결이 종료된다.
13	신규 Device 가 /oic/sec/acl2 Resource 로부터 ACL 프로비저닝 및 관리 서비스, rowneruuid property 를 찾고 DTLS 연결을 개방한다. 신규 Device 로부터 사용할 크리덴셜을 찾는다.
14 – 15	신규 Device 가 로컬 Resource 에의 액세스를 실시하는데 사용할 ACL Resource 를 취득한다.
16 – 18	신규 Device 가 즉시 또는 Device Resource request 에 응답해서 SACL Resource 를 취득한다.
19 – 20	신규 Device 가 액세스 제어를 결정하기 위해 Access Manager 를 참조해야 하는 Resource 의 목록을 취득한다.

21	신규 Device 가 ACL Resource 의 프로비저닝을 반영하도록 Cm 을 갱신한다.
22	DTLS 연결이 종료된다.

표 15 – 다중 지원 서비스를 포함하는 Server 지향 프로비저닝 단계

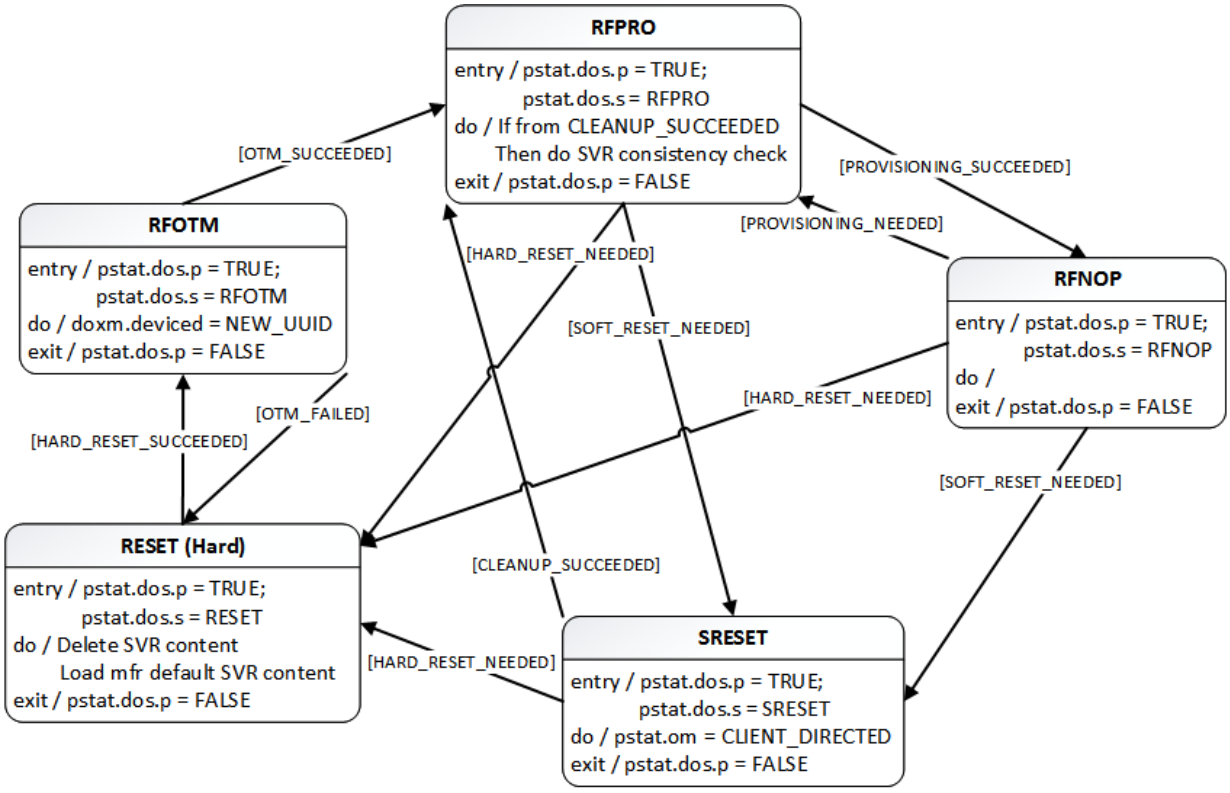
7.5 부트스트랩 예

이 섹션은 의도적으로 공백으로 남겨 두었음.

8 Device 온보딩 상태 정의

섹션 **Error! Reference source not found.**에 설명된 바와 같이 Device 의 소유권이 이전되고 섹션 **Error! Reference source not found.**에 설명된 바와 같이 Device 가 관련된 구성/서비스와 함께 프로비저닝되고 나면 온보딩 프로세스가 완료된다. 아래의 도면은 Device 의 생명 주기 동안에 Device 가 취할 수 있는 다양한 상태를 보여준다.

/pstat.dos.s property 는 /pstat resource 소유자 (예: 'doxs' 또는 'bss' 서비스)에 의해 RW 이므로 resource 소유자가 Device 상태를 원격으로 갱신할 수 있다. Device 의 상태가 RFNOP 또는 RFPRO 일 때, ACL 은 다른 Device 에 의해 Device 의 원격 제어가 가능하도록 하는데 사용될 수 있다. Device 의 상태가 SRESET 일 때는, Device 소유자 크리덴셜이 Device 에의 액세스를 허가하는 유일한 표시일 수 있다. Device 소유자는 Device 가 RFPRO 로 전환하는데 적합하도록 저 레벨 연속성 점검 및 재 프로비저닝을 수행할 수 있다.



도 28 – Device 상태 모델

도시된 바와 같이, 프로비저닝의 최종 단계에서 Device 는 다른 Device 와의 연동을 시작하는데 필요한 모든 것을 갖춘 "Ready for Normal Operation" 상태로 된다. 섹션 8.1 은 Device 가 "Ready for Normal Operation" 상태로 된 것으로 간주되기 위해 갖춰야 하는 최소한의 필수 구성을 규정한다.

전력이 손실되거나 Device 고장 시에, Device 는 전력 손실/고장 전의 상태를 그대로 유지하는 것이 좋다.

Device 또는 resource 소유자가 /pstat.dos.s 를 OBSERVE 하는 경우, SRESET 으로 전환하면 SVR 연속성 점검을 필요로 할 수 있는 Device 에 대한 조기 경보가 통지된다.

온보딩이 기능하도록 하려면, Device 에 다음과 같은 Resource 가 설치되어 있어야 한다.

1. /oic/sec/doxm Resource
2. /oic/sec/pstat Resource
3. /oic/sec/cred Resource

이들 Resource 에 포함된 값은 아래의 상태 정의에 규정된다.

8.1 Device 온보딩 리셋 상태 정의

/pstat.dos.s = RESET 상태는 제조사의 기본설정으로의 "하드" 리셋으로 정의된다. 하드 리셋은 Device 자산을 상대방에게 전달할 준비가 완료된 상태를 정의하기도 한다.

Platform 제조사는 강제로 Platform 을 리셋하기 위한 물리적 메커니즘 (예: 버튼)을 제공하는 것이 좋다. Platform 리셋이 실행되면 동일한 Platform 에 호스트된 모든 Device 가 Device 상태를 RESET 으로 전환한다.

다음의 Resource 와 이들의 고유 property 는 지정된 대로의 값을 갖는다.

1. /oic/sec/doxm Resource 의 "owned" Property 는 FALSE 로 전환된다.
2. /oic/sec/doxm Resource 의 "devowneruuid" Property 빈 UUID 를 갖는다.
3. /oic/sec/doxm Resource 의 "devowner" Property 는 구현되어 있는 경우 빈 UUID 를 갖는다.
4. /oic/sec/doxm Resource 의 "deviceuuid" Property 는 제조사의 기본설정 값으로 리셋된다.
5. /oic/sec/doxm Resource 의 "deviceid" Property 는 구현되어 있는 경우 제조사의 기본설정 값으로 리셋된다.
6. /oic/sec/doxm Resource 의 "sct" Property 는 제조사의 기본설정 값으로 리셋된다.

7. /oic/sec/doxm Resource 의 "oxmsel" Property 는 제조사의 기본설정 값으로 리셋된다.

8. /oic/sec/pstat Resource 의 "isop" Property 는 FALSE 이다.

9. /oic/sec/pstat Resource 의 "dos"는 갱신된다: dos.s 는 "RESET" 상태로 되고 dos.p 는 "FALSE"로 된다.

10. /oic/sec/pstat Resource 의 현재 프로비저닝 모드 Property - "cm"은 "00000001"이다.

11. /oic/sec/pstat Resource 의 목표 프로비저닝 모드 Property - "tm"은 "00000010"이다.

12. /oic/sec/pstat Resource 의 동작 모드 Property - "om"은 제조사의 기본설정 값으로 리셋된다.

13. /oic/sec/pstat Resource 의 지원되는 동작 모드 Property - "sm"은 제조사의 기본설정 값으로 리셋된다.

14. /oic/sec/pstat, /oic/sec/doxm, /oic/sec/acl, /oic/sec/amacl, /oic/sec/sacl, 및 /oic/sec/cred Resource 의 "rowneruuid" Property 는 빈 UUID 를 갖는다.

.

8.2 Device OTM 준비 완료 상태 정의

다음의 Resource 및 이들의 고유 property 는 소유권 이전 준비가 완료된 가동 Device 에 대해 지정된 대로의 값을 갖는다.

1. /oic/sec/doxm Resource 의 "owned" Property 는 FALSE 이며 TRUE 로 전환된다.

2. /oic/sec/doxm Resource 의 "devowner" Property 는 구현되어 있는 경우 빈 UUID 를 갖는다.

3. /oic/sec/doxm Resource 의 "devowneruuid" Property 는 빈 UUID 를 갖는다.

4. /oic/sec/doxm Resource 의 "deviceid" Property 는 구현되어 있는 경우 빈 UUID 를 갖는다. /oic/d 의 "di" Property 값은 정의되지 않는다.

5. /oic/sec/doxm Resource 의 "deviceuuid" Property 는 빈 UUID 를 가질 수 있다. /oic/d 의 "di" Property 값은 정의되지 않는다.

6. /oic/sec/pstat Resource 의 "isop" Property 는 FALSE 이다.

7. /oic/sec/pstat Resource 의 "dos"는 갱신된다: dos.s 는 "RFOTM" 상태로 되고 dos.p 는 "FALSE"로 된다.

8. /oic/sec/pstat Resource 의 "cm" Property 는 "00XXXX10"이다.

9. /oic/sec/pstat Resource 의 "tm" Property 는 "00XXXX00"이다.

10. /oic/sec/cred Resource 는 선택된 OTM 에 의해 요구되는 경우 크리덴셜을 포함한다.

8.3 Device Provisioning 준비 완료 상태 정의

Device 가 추가적인 프로비저닝 준비 완료 상태일 때, 다음의 Resource 및 이들 고유의 property 는 지정된 대로의 값을 갖는다.

1. /oic/sec/doxm Resource 의 "owned" Property 는 TRUE 이다.

2. /oic/sec/doxm Resource 의 "devowneruuid" Property 는 빈 UUID 가 아니어야 한다.

3. /oic/sec/doxm Resource 의 "deviceuuid" Property 는 빈 UUID 가 아니라 RFOTM 프로세싱 중에 결정된 값으로 설정되어야 한다. /oic/d Resource 의 "di" Property 값은 /oic/sec/doxm Resource 의 deviceid Property 값과 같아야 한다.

4. /oic/sec/doxm Resource 의 "oxmsel" Property 는 소유권 이전 시에 사용된 실제 OTM 과 동일한 값을 갖는다.

5. /oic/sec/pstat Resource 의 "isop" Property 는 FALSE 이다.

6. /oic/sec/pstat Resource 의 "dos"는 갱신된다: dos.s 는 "RFPRO" 상태로 되고 dos.p 는 "FALSE"로 된다.

7. /oic/sec/pstat Resource 의 "cm" Property 는 "00XXXX00"이다.

8. /oic/sec/pstat Resource 의 "tm" Property 는 "00XXXX00"이다.

9. 설치된 모든 Resource 의 "rowneruuid" Property 는 유효한 Resource 소유자 (즉, 주어진 Resource 를 인스턴스화하거나 갱신할 수 있도록 인가된 개체)로 설정된다. rowneruuid 또는 rowner (최소한 둘 중 하나)를 설정하지 못하면 orphan Resource 로 되는 경우가 있다.

10. /oic/sec/cred Resource 는 rowneruuid, amsuuid, 및 devowneruuid 에 의해 참조되는 각 개체의 크리덴셜을 포함한다.

8.4 Device 통상 동작 준비 완료 상태 정의

다음의 Resource 및 이들 고유의 property 는 가동 Device Final State 에 대해 지정된 대로의 값을 갖는다.

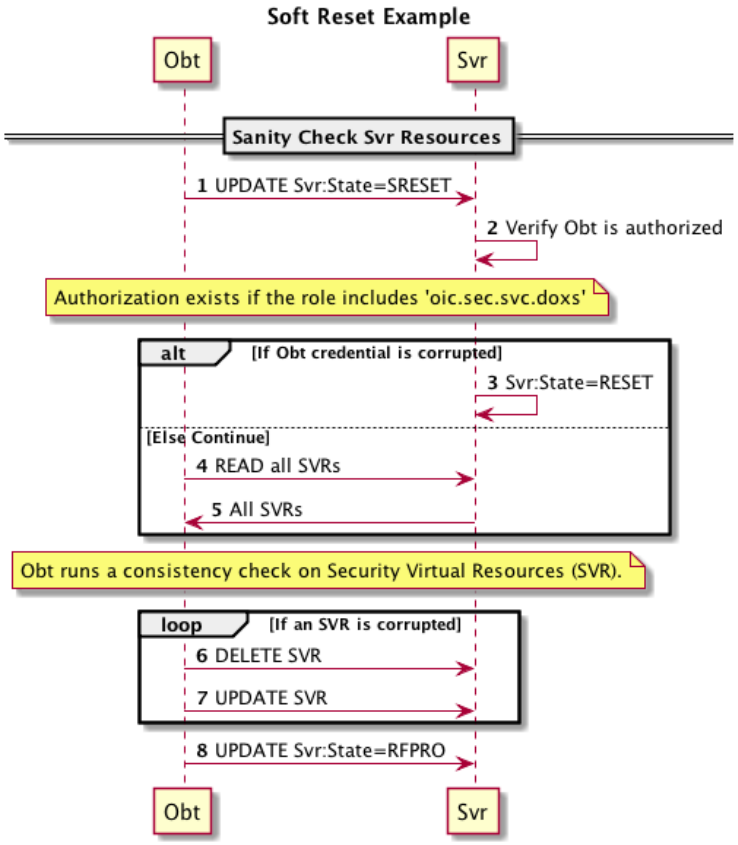
1. /oic/sec/doxm Resource 의 "owned" Property 는 TRUE 이다.
2. /oic/sec/doxm Resource 의 "devowneruuid" Property 는 빈 UUID 가 아니어야 한다.
3. /oic/sec/doxm Resource 의 "deviceuuid" Property 는 빈 UUID 가 아니라 OTM 시에 구성된 ID 로 설정되어야 한다. /oic/d 의 "di" Property 값은 deviceuuid 와 같아야 한다.

4. /oic/sec/doxm Resource 의 "oxmsel" Property 는 소유권 이전 시에 사용된 실제 OTM 과 동일한 값을 갖는다.
5. /oic/sec/pstat Resource 의 "isop" Property 는 TRUE 이다.
6. /oic/sec/pstat Resource 의 "dos"는 갱신된다: dos.s 는 "RFNOP" 상태로 되고 dos.p 는 "FALSE"로 된다.
7. /oic/sec/pstat Resource 의 "cm" Property 는 "00XXXX00"이다 (여기서, "X"는 1 또는 0 으로 해석된다).
8. /oic/sec/pstat Resource 의 "tm" Property 는 "00XXXX00"이다.
9. 설치된 모든 Resource 의 "rowneruuid" Property 는 유효한 resource 소유자 (즉, 주어진 Resource 를 인스턴스화하거나 갱신할 수 있도록 인가된 개체)로 설정된다. rowneruuid 또는 rowner (최소한 둘 중 하나)를 설정하지 못하면 orphan Resource 로 되는 경우가 있다.
10. /oic/sec/cred Resource 는 rowneruuid, amsuuid, 및 devowneruuid 에 의해 참조되는 각 서비스의 크리덴셜을 포함한다.

8.5 Device 소프트 리셋 상태 정의

소프트 리셋 상태는 (예: /pstat.dos.s = SRESET) Device 가 이 상태로 전환되면 동작하지 않지만 여전히 현재 소유자에 의해 소유된 상태임을 의미하는 경우를 정의한다. Device 원래 온보딩 시에 제공된 OC 를 사용해서 OBT 를 인증함으로써 (예: "rt" = "oic.r.doxs") SRESET 상태를 종료할 수 있다 (단, 소유자 이전 방법 /doxm.oxms 의 사용을 필요로 하지 않는다).

OBT 는 SVR 의 일관성 점검을 수행하고, 필요하면 Device 가 RFPRO 로 전환할 수 있도록 충분히 재프로비저닝해야 한다.



도 29 – SRESET 에서의 OBT Sanity 체크 시퀀스

OBT 는 최종적으로 RFPRO Device 상태로 전환하기 전에 SVR 의 정상 여부 확인을 수행해야 한다. Device 의 OBT 크리덴셜을 찾을 수 없거나 크리덴셜이 손상되었다고 판단되는 경우에는, Device 상태가 RESET 으로 전환된다. OBT 크리덴셜이 OBT 를 인증하지 못하면, Device 는 SRESET 상태를 유지해야 한다. 이렇게 함으로써 비 OBT Device 에 의해 시도되는 서비스 거부 공격을 완화할 수 있다.

SRESET 상태에서 다음의 Resource 및 이들의 고유 Property 는 지정된 대로의 값을 갖는다.

1. /oic/sec/doxm Resource 의 "owned" Property 는 TRUE 이다.
2. /oic/sec/doxm Resource 의 "devowneruuid" Property 는 null 이 아닌 값을 유지한다.
3. /oic/sec/doxm Resource 의 "devowner" Property 는 구현되어 있는 경우 null 이 아닌 값을 갖는다.

4. /oic/sec/doxm Resource 의 "deviceuuid" Property 는 null 이 아닌 값을 유지한다.
5. /oic/sec/doxm Resource 의 "deviceid" Property 는 null 이 아닌 값을 유지한다.
6. /oic/sec/doxm Resource 의 "sct" Property 는 그 값을 유지한다.
7. /oic/sec/doxm Resource 의 "oxmsel" Property 는 그 값을 유지한다.
8. /oic/sec/pstat Resource 의 "isop" Property 는 FALSE 이다.
9. /oic/sec/pstat.dos.s Property 는 SRESET 상태이다.
10. /oic/sec/pstat Resource 의 현재 프로비저닝 모드 Property - "cm"은 "00000001"이다.

11. /oic/sec/pstat Resource 의 목표 프로비저닝 모드 Property - "tm"은 "00XXXX00"이다.

12. /oic/sec/pstat Resource 의 동작 모드 Property - "om"는 'client-지향 mode'이다.

13. 지원되는 동작 모드 Property (/pstat.sm)는 Device 소유자 (aka DOXS)에 의해 갱신될 수 있다.

14. /oic/sec/pstat, /oic/sec/doxm, /oic/sec/acl, /oic/sec/acl2, /oic/sec/amacl, /oic/sec/sacl, 및 /oic/sec/cred Resource 의 "rowneruuid" Property 는 Device 소유자 (aka DOXS)에 의해 리셋되어 재 프로비저닝될 수 있다.

9 보안 크리덴셜 관리

이 섹션에서는 크리덴셜의 사용, 프로비저닝, 및 지속적인 관리와 더불어 OCF 에서의 크리덴셜 타입의 개요를 제공한다.

9.1 크리덴셜 생명 주기

OCF 크리덴셜 생명 주기는 다음과 같은 단계를 갖는다: (1) 생성, (2) 삭제, (3) 갱신, (4) 발행, 및 (5) 폐기.

9.1.1 생성

Device 는 Diffie-Hellman 과 같은 ad-hoc 키 교환 방법을 사용해서 크리덴셜 Resource 를 직접 인스턴스화할 수 있다. 또는, CMS 를 사용해서 Device 로 크리덴셜 Resource 를 프로비저닝할 수 있다.

크리덴셜 Resource 는 CMS 를 식별하는 resource owner property (/oic/sec/cred.Rowner)를 유지한다. 크리덴셜이 ad-hoc 을 통해 생성된 경우, Key Exchange 에 관여된 peer Device 가 CMS 로 간주된다.

CMS 를 사용해서 생성된 크리덴셜 Resource 는 전용 크리덴셜 발행 프로토콜 및 메시지를 포함할 수 있다. 이는 인증 기관 (CA)과 같은 public key infrastructure (PKI), key distribution centre (KDC)와 같은 또는 프로비저닝, 부트스트랩, 또는 온보딩 서비스의 일환으로서의 대칭 키 관리의 사용을 포함할 수 있다.

9.1.2 삭제

CMS 가 크리덴셜 Resource 를 삭제하거나 Device (예: 크리덴셜 Resource 가 호스트된 Device)가 직접 크리덴셜 Resource 를 삭제할 수 있다.

만료된 크리덴셜 Resource 는 메모리 및 저장 공간을 관리하기 위해 삭제될 수 있다.

OCF 키 관리에서의 삭제는 크리덴셜 정지와 동등하다.

9.1.3 갱신

크리덴셜의 갱신은 만료 전에 CMS 를 통해 수행할 수 있다.

처음에 크리덴셜을 취득하는데 사용된 방법이 크리덴셜의 갱신에 사용되어야 한다.

/oic/sec/cred Resource 는 Period Property 를 사용해서 크리덴셜의 만료를 지원한다. 크리덴셜의 갱신은 크리덴셜의 만료가 가깝거나 크리덴셜이 암호화 바이트의 최대 임계치를 초과하려고 할 때 적용할 수 있다.

크리덴셜 갱신 방법은 키 갱신 수행 시에 사용 가능한 옵션을 지정한다. 크리덴셜이 만료될 때는 Period Property 가 통지된다. Device 는 현재 만료되지 않은 크리덴셜을 사용해서 기존의 크리덴셜을 갱신함으로써 사전에 새로운 크리덴셜을 취득할 수 있다. Device 가 내장 타임 소스를 갖고 있지 않으면 CMS 로부터 정기적으로 현재 시각을 취득해야 한다.

또는, 신뢰할 수 있는 CMS 가 없지 않은 한, CMS 를 사용해서 만료된 크리덴셜을 갱신하거나 재 발행할 수 있다.

CMS 크리덴셜이 만료되면, BSS 또는 온보딩 서비스를 사용해서 CMS 를 재 프로비저닝할 수 있다. 온보딩에 의해 정해진 크리덴셜이 만료되면 Device 를 다시 온보딩하고 device 소유권 이전 단계를 다시 적용해야 한다.

ad-hoc 방법으로 정해진 크리덴셜이 만료되면 ad-hoc 방법을 다시 적용해야 한다.

모든 Device 는 최소한 하나의 크리덴셜 갱신 방법을 지원한다.

9.1.4 폐기

CMS 에 의해 발행된 크리덴셜에는 폐기 기능이 있는 경우가 있다. 폐기 방법이 정상적인 만료 기간 전에 폐기할 크리덴셜을 식별하는 폐기 객체의 프로비저닝을 포함하는 경우, 원래 발행된 크리덴셜을 대체할 폐기 정보를 포함하는 크리덴셜 Resource 가 생성된다. 만료 시에 폐기 객체가 삭제되도록 폐기 객체의 만료는 폐기된 크리덴셜과 일치해야 한다.

폐기는 크리덴셜 또는 Device 에의 적용을 고려하는 것이 개념적으로 타당하다. Device 폐기는 폐기된 Device 에 관련된 모든 크리덴셜이 폐기됨을 의미한다. Device 를 분실하거나, 도난 당하거나, 또는 훼손되었을 때 Device 를 폐기할 필요가 있다. 폐기된 Device 상의 크리덴셜의 삭제는 불가능하거나 신뢰할 수 없다.

9.2 크리덴셜 타입

/oic/sec/cred Resource 는 몇몇 암호 키와 인증 및 데이터 보호에 사용되는 그 밖의 정보를 지원하는 크리덴셜 타입 Property 를 보유한다. 지원되는 크리덴셜 타입은 쌍 대칭 키, 그룹 대칭 키, 비 대칭 인증 키, 인증서 (즉, 서명된 비 대칭 키), 및 공유 비밀 (즉, PIN/비밀번호)을 포함한다.

9.2.1 쌍 대칭 키 크리덴셜

쌍 대칭 키 크리덴셜은 정확히 하나의 다른 피어 Device 와 공통으로 대칭 키를 갖는다. CMS 는 대칭 키의 인스턴스를 보유할 수 있다. CMS 는 쌍을 이루는 키를 발행 또는 프로비저닝하고 쌍이 되는 피어의 하나로 가장하는데 오용하지 않는 것으로 신뢰할 수 있다.

쌍을 이루는 키는 ad-hoc 키 합의 프로토콜을 통해 구성할 수 있다.

/oic/sec/cred Resource 의 PrivateData Property 는 대칭 키를 포함한다.

PublicData Property 는 쌍을 이루는 키를 포함하는 피어 Device 에 대해 암호화된 토큰을 포함할 수 있다.

OptionalData Property 는 폐기 상태를 포함할 수 있다.

Device 구현자는 PrivateData 의 사적인 유지를 보장하는 강화된 키 저장 기술을 적용해야 한다.

Device 구현자는 무단 변경을 방지하기 위해 /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, 및 /oic/sec/csr Resource 에 대한 적합한 무결성, 기밀성, 및 액세스 보호를 적용해야 한다.

9.2.2 그룹 대칭 키 크리덴셜

그룹 키는 Device 의 그룹 (3 이상) 내에서 공유되는 대칭 키이다. 그룹 키는 그룹 참여자 간에 효율적인 데이터 공유를 위해 사용된다.

그룹 키는 Device 의 인증은 제공하지 않고 그룹 내의 멤버십만 설정한다.

그룹 키는 CMS 의 지원으로 배포된다. CMS 는 그룹 키를 발행 또는 프로비전하고 보호된 데이터를 조작하는데 이를 오용하지 않는 것으로 신뢰할 수 있다.

/oic/sec/cred Resource 의 PrivateData Property 는 대칭 키를 포함한다.

PublicData Property 는 그룹 명칭을 포함할 수 있다.

OptionalData Property 는 폐기 상태를 포함할 수 있다.

Device 구현자는 PrivateData 의 사적인 유지를 보장하는 강화된 키 저장 기술을 적용해야 한다.

Device 구현자는 무단 변경을 방지하기 위해 /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, 및 /oic/sec/csr Resource 에 대한 적합한 무결성, 기밀성, 및 액세스 보호를 적용해야 한다.

9.2.3 비 대칭 인증 키 크리덴셜

비 대칭 인증 키 크리덴셜은 공개 키 및 개인 키 쌍 또는 공개 키만을 포함한다. 개인 키는 Device 인증 시도 시의 서명에 사용된다. 공개 키는 device 인증 시도 응답을 검증하는데 사용된다.

/oic/sec/cred Resource 의 PrivateData Property 는 개인 키를 포함한다.

PublicData Property 는 공개 키를 포함한다.

OptionalData Property 는 폐기 상태를 포함할 수 있다.

Device 구현자는 PrivateData 의 사적인 유지를 보장하는 강화된 키 저장 기술을 적용해야 한다.

Device 는 개인 키가 Device 에 의해서만 알 수 있도록 내부적으로 비 대칭 인증키 쌍을 생성해야 한다. 언제 Device 간에 개인 키 재료를 전송할 필요가 있는지에 관해서는 섹션 9.2.3.1 을 참조하기 바란다.

Device 구현자는 무단 변경을 방지하기 위해 /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, 및 /oic/sec/csr Resource 에 대한 적합한 무결성, 기밀성, 및 액세스 보호를 적용해야 한다.

9.2.3.1 비 대칭 인증 키 크리덴셜의 외부 생성

device 는 device 외부에서의 키 쌍 생성 및 프로비저닝을 허용할 때 산업 표준의 보증된 기술을 사용해야 한다. 특히 키 쌍이 프로비저닝 후에 변경 또는 대체할 수 없는 경우에는 그러한 키 쌍의 사용을 최소화해야 한다.

온보딩의 일환으로 사용 시에, 이러한 키 쌍은 OBT 또는 사용자가 Device 의 온보딩 수용을 확신할 수 있도록 Device 가 인증서에 제조사 주장 property 를 소유하고 있음을 증명하는데 사용된다. Device 의 인증에 그러한 인증서를 사용하고 사용할 신규 네트워크 크리덴셜을 프로비저닝하는 소유권 이전 방법에 관해서는 섹션 **Error! Reference source not found.**을 참조하기 바란다.

9.2.4 비 대칭 키 암호 키 크리덴셜

비 대칭 키 암호 키 (KEK) 크리덴셜은 키를 분배 또는 저장할 때 대칭 키를 래핑하는데 사용된다.

/oic/sec/cred Resource 의 PrivateData Property 는 개인 키를 포함한다.

PublicData Property 는 공개 키를 포함한다.

OptionalData Property 는 폐기 상태를 포함할 수 있다.

Device 구현자는 PrivateData 의 사적인 유지를 보장하는 강화된 키 저장 기술을 적용해야 한다.

Device 구현자는 무단 변경을 방지하기 위해 /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, 및 /oic/sec/csr Resource 에 대한 적합한 무결성, 기밀성, 및 액세스 보호를 적용해야 한다.

9.2.5 인증서 크리덴셜

인증서 크리덴셜은 CMS 또는 외부 인증 기관 (CA)에 의해 발행된 인증서에 동반되는 비 대칭 키이다.

인증 등록 프로토콜은 인증서를 취득하고 소유 증명을 설정하는데 사용된다.

발행된 인증서는 비 대칭 키 크리덴셜 Resource 와 함께 저장된다.

인증서 폐기 상태와 같은 인증서 생명 주기를 관리하는데 유용한 그 밖의 객체는 크리덴셜 Resource 에 관련 지어진다.

비 대칭 키 크리덴셜 Resource 또는 자체 서명 인증서 크리덴셜이 경로 검증을 종료하는데 사용된다.

/oic/sec/cred Resource 의 PrivateData Property 는 개인 키를 포함한다.

PublicData Property 는 발행된 인증서를 포함한다.

OptionalData Property 는 폐기 상태를 포함할 수 있다.

Device 구현자는 PrivateData 의 사적인 유지를 보장하는 강화된 키 저장 기술을 적용해야 한다.

Device 구현자는 무단 변경을 방지하기 위해 /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, 및 /oic/sec/csr Resource 에 대한 적합한 무결성, 기밀성, 및 액세스 보호를 적용해야 한다.

9.2.6 비밀번호 크리덴셜

공유 비밀 크리덴셜은 외부 시스템 또는 OCF 크리덴셜 타입을 지원하지 않는 Device 에의 Device 액세스를 허가하기 위한 PIN 또는 비밀번호를 보유하는데 사용된다.

/oic/sec/cred Resource 의 PrivateData Property 는 PIN, 비밀번호, 및 비밀번호의 변경 및 검증에 사용되는 그 밖의 값을 포함한다.

PublicData Property 는 해당하는 경우 사용자 또는 계정의 명칭을 포함할 수 있다.

OptionalData Property 는 폐기 상태를 포함할 수 있다.

Device 구현자는 PrivateData 의 사적인 유지를 보장하는 강화된 키 저장 기술을 적용해야 한다.

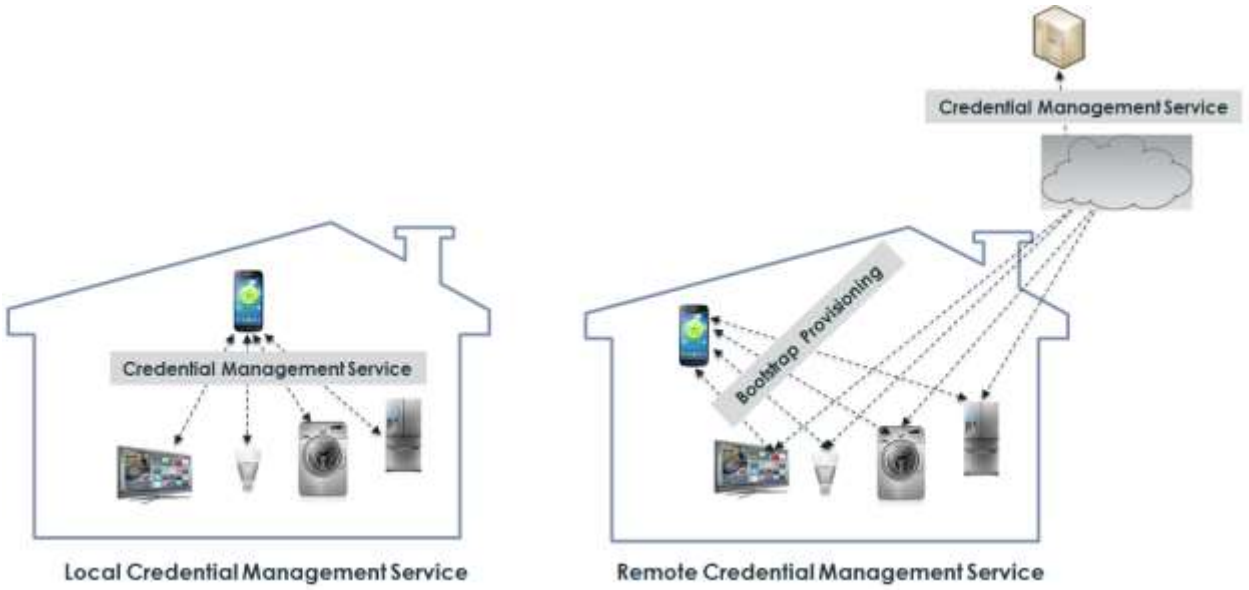
Device 구현자는 무단 변경을 방지하기 위해 /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, 및 /oic/sec/csr Resource 에 대한 적합한 무결성, 기밀성, 및 액세스 보호를 적용해야 한다.

9.3 Certificate 기반 키 관리

9.3.1 개요

OCF 네트워크 내에서 통신 시에 인증을 획득하고 보안 전송을 수행하기 위해 통신 당사자의 공개 키를 포함하는 인증서 및 개인 키를 사용할 수 있다.

Device 가 OCF 네트워크 내에서 전개되고 CMS (Credential Management Service)에 의해 크리덴셜 프로비저닝이 지원되면, 인증서와 개인 키는 로컬 또는 원격 인증 기관 (CA)에 의해 발행될 수 있다. 로컬 CA 의 경우, X.509 를 기반으로 한 인증서 폐기 목록 (CRL)을 사용해서 신원 증명을 확인할 수 있다. 원격 CA 의 경우, Online Certificate Status Protocol (OCSP)을 사용해서 신원 증명 및 유효성을 확인할 수 있다.



도 30 – Certificate 관리 아키텍처

OCF 인증서 및 OCF CRL (Certificate Revocation List) 형식은 X.509 형식의 부분집합으로, elliptic curve algorithm 및 DER 인코딩 형식만 허용되고, X.509 의 대부분의 옵션 필드가 지원되지 않아서, 제약된 Device 의 요구 사항을 만족시키고자 의도하고 있다.

Server 에서의 인증서 및 CRL 관리에 있어서, 인증서 및 CRL 의 Resource 의 저장, 검색, 및 파싱은 보안 resource 매니저 계층에서 수행되고, 관련된 Interface 는 상위 계층으로 노출될 수 있다.

SRM 은 섹션 5.4 에 설명된 바와 같이 Server 에서의 보안 시행점이므로 인증서 및 CRL 의 데이터는 SVR 데이터베이스에서 저장 및 관리된다.

Device 가 새로 온보딩되거나 Device 의 인증서가 폐기되었을 때 Device 의 인증서 발행 요청은 CMS 에 의해 처리되어야 한다. 인증서가 무효하다고 여겨지면 인증서를 폐기해야 한다. CRL 은 폐기된 인증서와 신뢰할 수 없는 해당 Device 의 목록을 포함하는 데이터 구조이다. 실제 운영에서 CRL 은 정기적으로 (예를 들어, 3 개월마다) 갱신될 것으로 예상된다.

9.3.2 인증서 형식

OCF 의 인증서 형식은 [RFC5280]에 정의된 X.509 형식 (버전 3 이상)의 부분집합이다.

9.3.2.1 Certificate 프로파일 및 필드

OCF 인증서는 다음과 같은 필드를 지원한다: version, serialNumber, signature, issuer, validity, subject, subjectPublicKeyInfo, extensions, signatureAlgorithm, 및 signatureValue.

- version: 인코딩된 인증서의 버전
- serialNumber: 인증서 일련번호
- signature: CA 에 의해 인증서 서명에 사용된 알고리즘의 알고리즘 식별자
- issuer: 인증서에 서명하고 인증서를 발행한 개체
- validity: CA 가 보증하는 기간
- subject: 주체 공개 키 필드 (deviceID)에 관련된 개체
- subjectPublicKeyInfo: 공개 키 및 키가 함께 사용되는 알고리즘
- extensions: 섹션 9.3.2.2 에 정의된 인증서 확장자
- signatureAlgorithm: CA 에 의해 인증서 서명에 사용된 암호화 알고리즘
- signatureValue: ASN.1 DER 인코딩된 OCFtbsCertificate 에 대해 산출된 디지털 서명 (이 서명 값은 BIT STRING 으로 인코딩된다)

OCF 인증서 구문은 다음과 같이 정의된다.

```
OCFCertificate ::= SEQUENCE {
        OCFtbsCertificate       TBSCertificate,
        signatureAlgorithm      AlgorithmIdentifier,
        signatureValue          BIT STRING
}
```

OCFtbsCertificate 필드는 주체 및 발행자의 명칭, 주체에 관련된 공개 키, 유효 기간, 및 그 밖의 관련 정보를 포함한다. RFC5280 에 따라, 버전 3 인증서는 버전 필드에 값 2 를 사용하여 버전 번호를 인코딩한다. 아래의 문법은 버전 2 인증서를 허용하지 않는다.


```
OCFtbsCertificate  ::=  SEQUENCE  {
        version            [0]  2 or above,
        serialNumber       CertificateSerialNumber,
        signature          AlgorithmIdentifier,
        issuer             Name,
        validity           Validity,
        subject            Name,
       subjectPublicKeyInfo  SubjectPublicKeyInfo,
        extensions     [3] EXPLICIT Extensions
}
subjectPublicKeyInfo  ::=  SEQUENCE  {
        algorithm             AlgorithmIdentifier,
        subjectPublicKey      BIT STRING
}
Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
       extnID      OBJECT IDENTIFIER,
       critical    BOOLEAN DEFAULT FALSE,
       extnValue   OCTET STRING
                   -- contains the DER encoding of an ASN.1 value
                   -- corresponding to the extension type identified
                   -- by extnID
}
```

Certificate 필드		설명	OCF	X.509
OCFtbsCertificate	version	2 이상	필수	필수
	serialNumber	CertificateSerialNumber	필수	필수
	signature	AlgorithmIdentifier	1.2.840.10045.4.3.2 (SHA256 를 가진 ECDSA 알고리즘, 필수)	[RFC3279], [RFC4055], 및 [RFC4491]에 규정
	issuer	명칭	필수	필수
	validity	유효성	필수	필수
	subject	명칭	필수	필수
	subjectPublicKeyInfo	SubjectPublicKeyInfo	1.2.840.10045.2.1,1.2.840.10045.3.1.7	[RFC3279], [RFC4055], 및

			(secp256r1 커브를 토대로 한 SHA256 를 가진 ECDSA 알고리즘, 필수)	[RFC4491]에 규정
	issuerUniqueID	암시적 UniqueIdentifier	지원되지 않음	선택
	subjectUniqueID	암시적 UniqueIdentifier	지원되지 않음	
	extensions	명시적 확장자	필수	
signatureAlgorithm		AlgorithmIdentifier	1.2.840.10045.4.3.2 (SHA256 를 가진 ECDSA 알고리즘, 필수)	[RFC3279], [RFC4055], 및 [RFC4491]에 규정
signatureValue		BIT STRING	필수	필수

표 16 – OCF 와 X.509 간의 certificate 필드 비교

9.3.2.2 지원되는 인증서 확장자

이들 인증서 확장자는 RFC 5280 의 표준 부분이므로, 본 스펙에서는 RFC 의 섹션 번호를 참조로 포함한다. 아래에 각 확장자를 요약하고 RFC 정의에 대한 수정을 나열한다. Device 는 여기에 나열된 확장자를 구현하고 이해해야 한다. 본 스펙에 포함되지 않은 그 밖의 RFC 확장자는 필요하지 않다. 섹션 10.3 에서는 확장자의 처리를 포함해서 인증서 체인 검증 시에 어떤 Device 가 구현해야 하고 특정 확장자가 없을 때 어떤 동작을 취해야 하는지를 설명한다.

- 인증 기관 키 식별자 (4.2.1.1)

  인증 기관 키 식별자 (Authority Key Identifier: AKI) 확장자는 인증서의 서명에 사용된 개인 키에 대응하는 공개 키를 식별하기 위한 수단을 제공한다. 본 스펙에서는 이 확장자의 참조된 정의에 다음과 같은 수정을 가한다.

  AuthorityKeyIdentifier 시퀀스의 authorityCertIssuer 또는 authorityCertSerialNumber 필드는 허용되지 않고, keyIdentifier 만 허용된다. 이는 다음과 같은 문법 정의로 이어진다.

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


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

KeyIdentifier ::= OCTET STRING
```

- 주체 키 식별자 (4.2.1.2)

주체 키 식별자 (Subject Key Identifier: SKI) 확장자는 특정 공개 키를 포함하는 인증서를 식별하기 위한 수단을 제공한다.

본 스펙에서는 이 확장자의 참조된 정의에 다음과 같은 수정을 가한다.

주체 키 식별자 (Subject Key Identifier)는 인증서의 SubjectPublicKeyInfo 필드에 포함된 공개 키로부터 도출되거나 고유한 값을 생성하는 방법으로 도출되어야 한다. 본 스펙에서는 BIT STRING subjectPublicKey 값의 256 비트 SHA-2 해시를 권장한다 (태그, 길이, 및 비사용 비트 수 제외). 인증서 체인을 검증하는 Device 는 키 식별자를 산출하기 위한 어떠한 특정 방법을 전제로 해서는 안되지만, 인증서에 보이는 키 식별자에 대한 인증 경로에서 AKI 와 SKI 의 매칭을 근거로 해야 한다.

- 주체 대체 이름

EKU 확장자가 존재하고 값 XXXXXX 를 갖고 있어서 role 인증서임을 가리킬 때는, 주체 대체 이름 (Subject Alternative Name) (subjectAltName)이 존재하고, 이는 다음과 같이 해석된다. EKU 가 없거나 다른 값을 갖고 있으면, subjectAltName 확장자가 존재하지 않아야 한다. subjectAltName 확장자는 role 인증서에서 하나 이상의 Role ID 값을 인코딩하는데 사용되어 주체 공개 키에 role 을 바인딩한다. subjectAltName 확장자는 RFC 5280 (섹션 4.2.1.6)에 정의된다.

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

SubjectAltName ::= GeneralNames

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

GeneralName ::= CHOICE {
    otherName                 [0]    OtherName,
```



```
        rfc822Name                  [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 }
```

role 을 인코딩하는 GeneralNames SEQUENCE 내의 각 GeneralName 은 Name 타입인 directoryName 이다. Name 은 X.501 고유의 Name 이다. 각 Name 은 정확히 하나의 CN (Common Name) 성분과 영 또는 하나의 OU (Organizational Unit) 성분을 갖는다. OU 성분은, 존재하는 경우, role 의 구문을 정의한 기관을 지정한다. OU 성분이 없으면, 인증서 발행자가 role 을 정의한 것이다. CN 성분은 role ID 를 인코딩한다. 그 밖의 GeneralName 타입이 SEQUENCE 내에 존재할 수 있지만 role 로 해석되지는 않는다. 그러므로, 인증서 발행자가 subjectAltName 확장자에 비 role 명칭을 포함하면, 확장자"중요한"으로 표기되어서는 안된다.

role 과 기관은 ASN.1 PrintableString 타입, 제한된 문자 집합 [0-9a-z-A-z '()+,-./:=?]으로 인코딩되어야 한다.

- 키 용법 (4.2.1.3)

키 용법 (Key Usage) 확장자는 인증서에 포함된 키의 목적을 정의한다 (예: 암호화, 서명, 인증서 서명). 복수의 동작에 사용될 수 있는 키를 제한할 때는 용법 제한이 적용될 수 있다.

본 스펙에서는 이 확장자의 참조된 정의에 수정을 가하지 않는다.

- 기본 제약 (4.2.1.9)

기본 제약 (Basic Constraints) 확장자는 인증서의 주체가 CA 인지를 식별하고 인증서를 포함하는 유효 인증 경로의 최대 깊이를 식별한다. 이 확장자가 없으면 인증서는 다른 인증서의 발행자가 될 수 없다.

본 스펙에서는 이 확장자의 참조된 정의에 수정을 가하지 않는다.

- 확장 키 용법 (4.2.1.12)

확장 키 용법 (Extended Key Usage) 확장자는 인증된 공개 키가 사용될 수 있는 허용된 목적을 기술한다. Device 가 인증서를 수신하면, 인증서가 존재하는 상호 작용의 컨텍스트를 토대로 목적을 파악하고 해당하는 목적에 인증서를 사용할 수 있는지 검증한다.

본 스펙에서는 이 확장자의 참조된 정의에 다음과 같은 수정을 가한다.

CA 는 이 확장자를 "중요한"으로 표기해야 한다.

CA 는 anyExtendedKeyUsage OID (2.5.29.37.0)로 인증서를 발행해서는 안된다.

OCF 고유의 목적 및 이를 표현하기 위해 할당된 OID 의 목록은 다음과 같다.

- 아이덴티티 인증서 1.3.6.1.4.1.44924.1.6
- Role 인증서 1.3.6.1.4.1.44924.1.7

9.3.2.3 인증, 크리덴셜, 및 무결성을 위한 Cipher Suite

인증서 기반 키 관리를 지원하는 모든 Device 는 [RFC7251]에 정의된 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suite 를 지원한다. 두 Device 간의 보안 채널을 구성하는 데는 ECDHE_ECDSA (즉, Diffie-Hellman 키 일치의 서명 버전) 키 합의 프로토콜이 사용된다. 이 프로토콜 사용 중에 양쪽이 서로 인증하게 된다. 데이터 전송의 기밀성은 AES_128_CCM_8 에 의해 제공된다. 데이터 전송의 무결성은 SHA256 에 의해 제공된다. 자세한 사항은 [RFC7251]에 정의된다.

최적화된 인증서 처리를 위해, 다음과 같이 필드의 값이 선택된다.

- signatureAlgorithm := ANSI X9.62 ECDSA algorithm with SHA256,

- signature := ANSI X9.62 ECDSA algorithm with SHA256,
- subjectPublicKeyInfo := ANSI X9.62 ECDSA algorithm with SHA256 based on secp256r1 curve.

인증서 유효 기간은 인증서의 상태에 관한 정보가 유지되는 것을 CA 가 보증하는 기간으로, 이 정보 필드는 두 날짜의 SEQUENCE 로 표현된다.

- 인증서 유효 기간의 시작 날짜 (notBefore)
- 인증서 유효 기간의 종료 날짜 (notAfter)

notBefore 와 notAfter 는 둘 다 UTCTime 으로 인코딩되어야 한다.

필드 발행자 및 주체는 인증서에 서명하고 인증서를 발행한 개체와 인증서의 소유자를 식별한다. 이들은 UTF8String 으로 인코딩되어 CN 속성에 삽입된다.

9.3.2.4 인증서의 인코딩

인증서의 인코딩에는 [ISO/IEC 8825-1]에 정의된 ASN.1 고유 인코딩 규칙 (DER)이 사용된다.

9.3.3 CRL 형식

OCF CRL 형식은 [RFC5280]을 토대로 하지만 선택 필드는 지원되지 않고 서명 관련 필드는 선택 사항이다.

9.3.3.1 CRL 프로파일 및 필드

OCF CRL 은 다음과 같은 필드를 지원한다: signature, issuer, this Update, revocationDate, signaturealgorithm, 및 signatureValue

- signature: CA 에 의해 CRL 서명에 사용된 알고리즘의 알고리즘 식별자
- issuer : CRL 에 서명하고 CRL 을 발행한 개체
- this Update : CRL 의 발행일
- userCertificate : 인증서 일련번호
- revocationDate : 폐기 일시

- signatureAlgorithm: CA 에 의해 CRL 서명에 사용된 암호화 알고리즘
- signatureValue: ASN.1 DER 인코딩된 OCFtbsCertificate 에 대해 산출된 디지털 서명 (이 서명 값은 BIT STRING 으로 인코딩된다)

signature, signatureAlgorithm, 및 signatureValue 와 같은 서명 관련 필드는 선택 사항이다.

```
CertificateList  ::=  SEQUENCE  {
        OCFtbsCertList         TBSCertList,
         signatureAlgorithm    AlgorithmIdentifier,
         signatureValue        BIT STRING
 }
OCFtbsCertList:: = SEQUENCE {
  signature            AlgorithmIdentifier OPTIONAL,
    issuer             Name,
    this Update        Time,
    revokedCertificates RevokedCertificates,
    signatureAlgorithm  AlgorithmIdentifier OPTIONAL,
    signatureValue      BIT STRING OPTIONAL
}
RevokedCertificates     SEQUENCE OF SEQUENCE  {
    userCertificate     CertificateSerialNumber,
    revocationDate      Time
}
```

CRL 필드			설명	OCF	X.509
OCFtbsCertList	version		버전 v2	지원되지 않음	선택
	signature		AlgorithmIdentifier	1.2.840.10045.4.3.2(SHA256 를 가진 ECDSA 알고리즘, 선택)	[RFC3279], [RFC4055], 및 [RFC4491] 목록 OID 에 규정
	issuer		명칭	필수	필수
	thisUpdate		시간	필수	필수
	nextUpdate		시간	지원되지 않음	선택
	revokedCertificates	userCertificate	인증서 일련번호	필수	필수
		revocationDa	시간	필수	필수

		te			
		crlEntryExtensions	시간	지원되지 않음	선택
	crlExtensions		확장자	지원되지 않음	선택
signatureAlgorithm			AlgorithmIdentifier	1.2.840.10045.4.3.2(SHA256 를 가진 ECDSA 알고리즘, 선택)	[RFC3279], [RFC4055], 및 [RFC4491] 목록 OID 에 규정
signatureValue			BIT STRING	선택	필수

표 17 – OCF 와 X.509 간의 CRL 필드 비교

9.3.3.2 CRL 의 인코딩

CRL 의 인코딩에는 [ISO/IEC 8825-1]에 정의된 ASN.1 고유 인코딩 규칙 (DER 인코딩 방법)이 사용된다.

9.3.4 Resource Model

Device 인증서와 개인 키는 cred Resource 에 보관된다. CRL 은 폐기 목록을 유지하도록 정의된 별도의 crl Resource 를 사용해서 유지 및 갱신된다.

cred Resource 는 Device 에 관련된 인증서 정보를 포함한다. PublicData Property 는 device 인증서와 CA 인증서 체인을 보유한다. PrivateData Property 는 인증서에 쌍으로 엮어진 Device 개인 키를 보유한다. (/oic/sec/cred Resource 에 관한 더 자세한 사항은 섹션 13.2 를 참조하기 바란다.)

인증서 폐기 목록 Resource 는 CMS 를 통해 취득된 폐기된 인증서의 목록을 유지하는데 사용된다. Device 는 폐기된 인증서를 인증서 경로 검증의 일환으로 고려해야 한다. CRL Resource 가 오래되거나 Platform Resource 가 전체 목록을 유지하는데 충분하지 않으면, Device 는 CMS 에 현재 폐기 상태를 질의해야 한다. (/oic/sec/crl Resource 에 관한 더 자세한 사항은 섹션 **Error! Reference source not found.**을 참조하기 바란다.)

9.3.5 인증서 프로비저닝

CMS (예: 허브 또는 스마트폰)는 신규 Device 에 대해 인증서를 발행한다. CMS 는 자신의 인증서 및 키 쌍을 갖는다. 인증서는 a) self-signed if it acts as Root CA 로 동작하는 경우 자체 서명된 것이거나 b) Sub CA 로 동작하는 경우 신뢰 계층에서 상위 CA 에 의해 서명된 것이다. 어느 경우에도 인증서는 섹션 9.3.2 에 기술된 형식을 가져야 한다.

CMS 의 CA 는 Device 의 공개 키와 개인 키 소유 증명을 검색하고, 이 CA 인증서에 의해 서명된 Device 의 인증서를 생성한다. 그리고 나서, CMS 는 CA 인증서 체인을 포함해서 Device 로 인증서를 전달한다. 선택적으로, CMS 는 섹션 9.3.2 에 기술된 형식을 갖는 하나 이상의 role 인증서도 전달하는 경우가 있다. 각 role 인증서의 subjectPublicKey 는 Device 인증서의 subjectPublicKey 와 일치해야 한다.

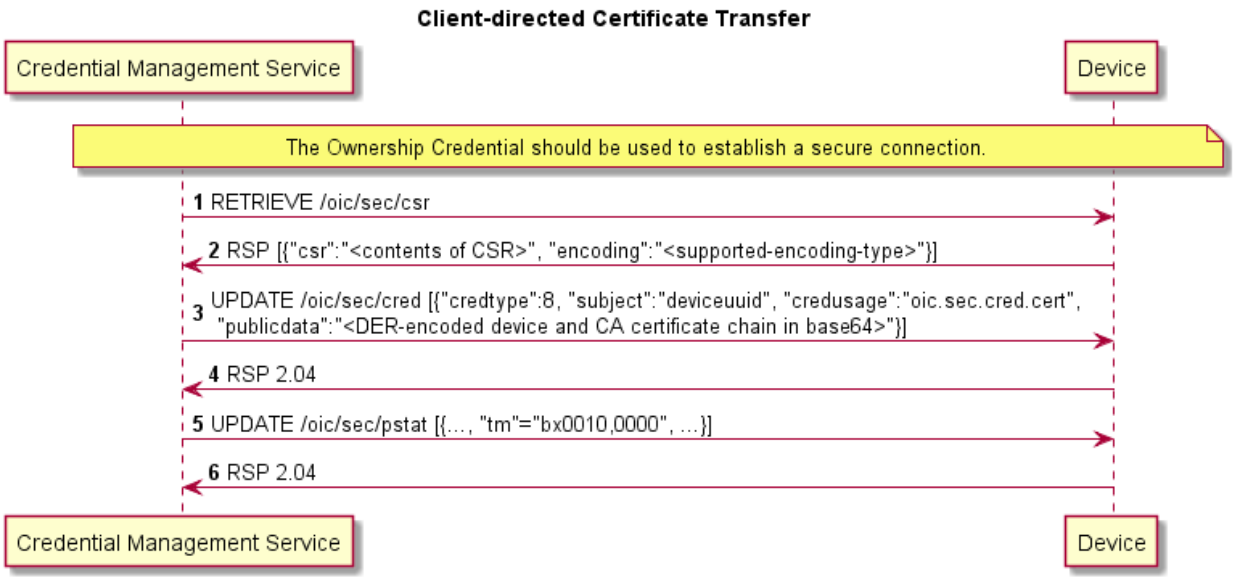
아래의 시퀀스에서 Certificate Signing Request (CSR)는 RFC 2986 의 PKCS#10 에 정의되고 여기에 참조된다.

Client 지향 모델에 대한 인증서 전달의 흐름을 도 31 에 도시한다.

1. CMS 가 인증서를 요청하는 Device 로부터 CSR 을 검색한다. 이 CSR 에서, Device 는 요청하는 UUID 를 주체에 기입하고 공개 키를 SubjectPublicKeyInfo 에 기입한다. Device 는 공개 키의 존재를 파악한다. 이것은 인증에 사용하기 위해 선택한 기존의 프로비저닝된 것일 수 있으며, 키가 존재하지 않으면, 내부적으로 신규 키 쌍을 생성하여 공개 부분을 제공한다. 키 쌍은 인증서가 OCF 인증에의 사용으로 제한되므로 섹션 9.3.2.3 및 **Error! Reference source not found.**에 나열된 허용 ciphersuites 에 대응해야 한다.

   Device 가 사전에 프로비저닝된 키 쌍을 가지고 있지 않고 키 쌍을 생성할 수 없으면, 인증서를 사용할 수 없다. 이 경우, Device 는 /oic/sec/doxm 의 sct property 내의 0x8 비트 위치를 0 으로 설정함으로써 이러한 사실을 알리고, /oic/sec/csr resource 가 존재하지 않는다는 오류를 리턴한다.

2. CMS 가 동일한 credid 를 사용해서 발행된 인증서와 CA 체인을 지정된 Device 로 전달해서 개인 키와의 관련을 유지한다. CMS 로부터 Device 로의 POST 내에 복수의 크리덴셜을 포함함으로써, 도 31 에서 인증서의 전달에 사용된 크리덴셜 타입 (oic.sec.cred)이 role 인증서의 전달에도 사용된다. 아이덴티티 인증서는 credusage property 를 `oic.sec.cred.cert'로 설정해서 저장되고, role 인증서는 credusage property 를 `oic.sec.cred.rolecert'로 설정해서 저장된다.



도 31 – Client 지향 인증서 전달

9.3.6 CRL Provisioning

CRL 발행을 위한 유일한 사전 요구 사항은 CMS (예: 허브 또는 스마트폰)가 폐기 인증서를 등록하고, CRL 에 서명하고 Device 로 전달하는 기능을 갖는 것이다.

CMS 는 Device 로 CRL 을 전달한다.

아래의 인증서 폐기 이유는 각 Device 에서 CRL 갱신의 원인이 된다.

- 발행자 명칭의 변경
- Device 와 CA 간의 관련 변경
- 인증서 훼손
- 대응하는 개인 키의 훼손 의심

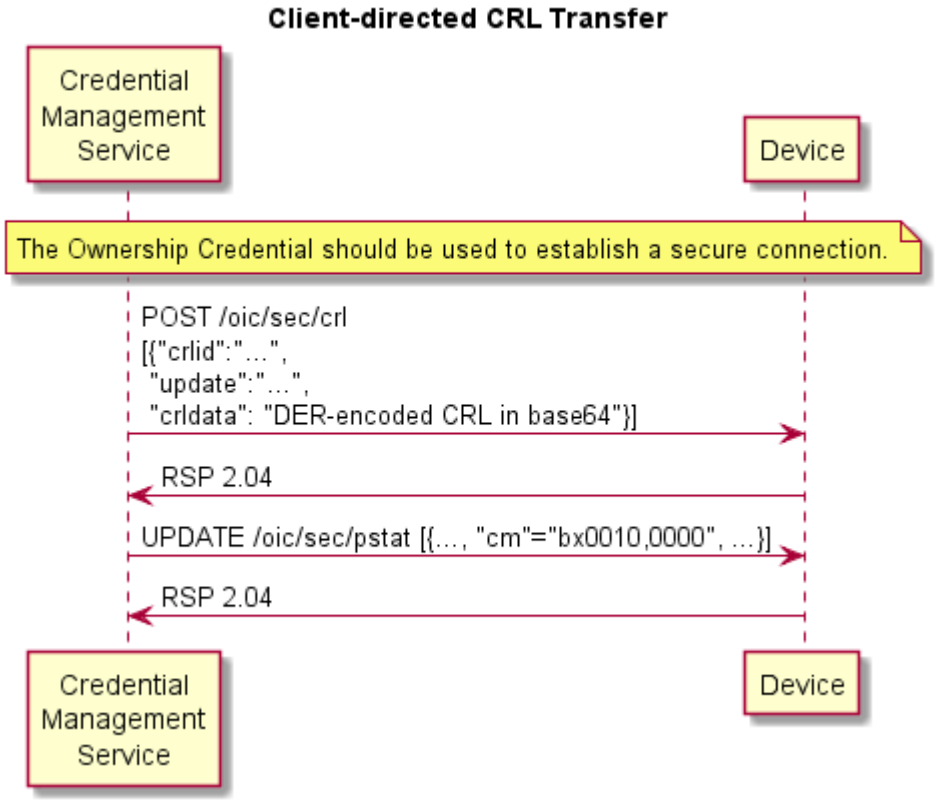
CRL 은 갱신되어 OCF 네트워크의 모든 액세스 가능한 Device 로 전달될 수 있다. 특별한 경우에, Device 는 주어진 CMS 에 CRL 을 요청할 수 있다.

CRL 을 갱신하고 전달하는 데는 두 가지 옵션이 있다.

- CMS 가 각 Device 에 CRL 을 일방적으로 전달한다.
- 각 Device 가 주기적으로 CRL 의 갱신을 요청한다.

Client 지향 모델에 대한 CRL 전달의 흐름을 도 32 에 도시한다.

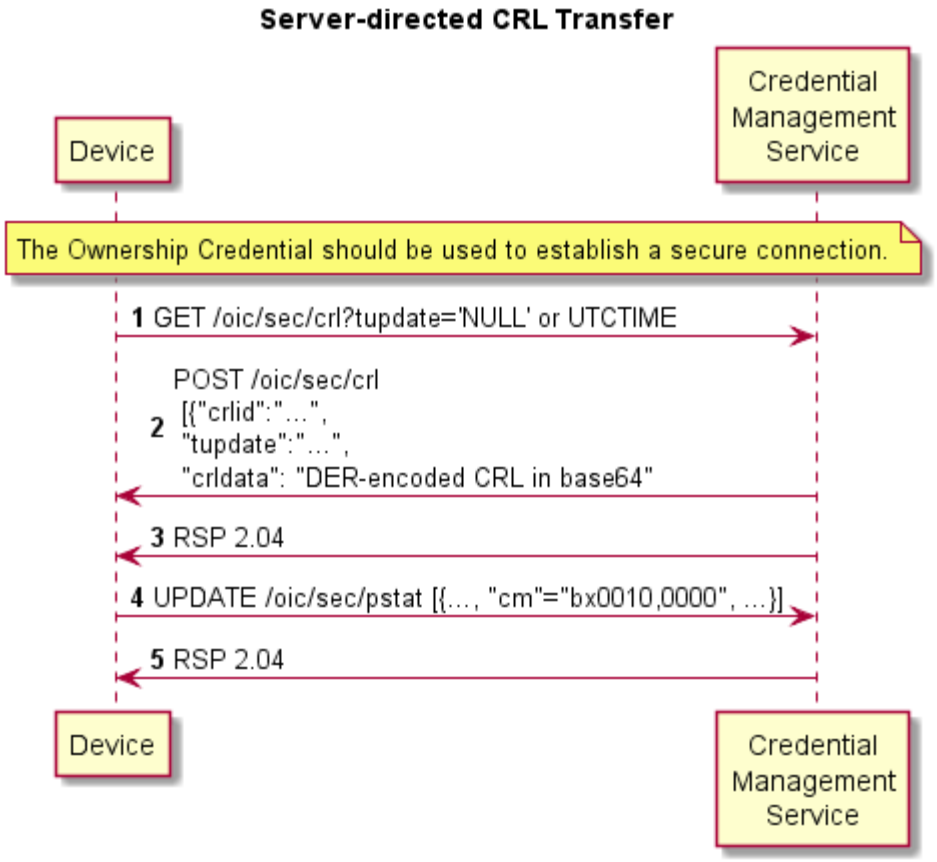
1. CMS 가 CRL Resource Property 를 검색한다.
2. Device 가 CMS 에 CRL 의 전달을 요청하면, CMS 가 Device 로 가장 최근의 CRL 을 전달한다 .



도 32 – Client 지향 CRL 전달

Server 지향 모델에 대한 CRL 전달의 흐름을 도 33 에 도시한다.

1. Device 가 CMS 에 대해 CRL Resource Property tupdate 를 검색한다.
2. CMS 가 지정된 tupdate 시간 후에 갱신된 CRL 정보를 인식하면, Device 로 CRL 을 전달한다.



2360 **도 33 – Server 지향 CRL 전달**

2361

10 Device 인증

Client 가 Server 상의 제한된 Resource 에 액세스할 때, Server 는 Client 를 인증한다. 액세스의 요청 시에 Client 는 Server 를 인증한다. Client 는 server 가 액세스 제어 결정에 사용할 수 있는 하나 이상의 role 을 주장할 수도 있다.

10.1 대칭 키 크리덴셜을 사용한 Device 인증

인증에 대칭 키를 사용할 때, Server Device 는 ServerKeyExchange 메시지를 포함하고 psk_identity_hint 를 Server 의 Device ID 로 설정한다. Client 는 Subject ID 가 Server 의 Device ID 로 설정된 크리덴셜을 갖고 있고 크리덴셜 타입이 PSK 인 것을 확인한다. 그렇지 않으면 Client unknown_psk_identity 에러 또는 그 밖의 적절한 에러로 응답한다.

Client 가 적절한 PSK 크리덴셜을 찾으면, Client 의 Device ID 로 설정된 psk_identity_hint 를 포함하는 ClientKeyExchange 메시지로 응답한다. Server 는 일치하는 Subject ID 와 타입을 포함하는 크리덴셜을 갖고 있는지 확인한다. 그렇지 않으면, Server 는 unknown_psk_identity 또는 그 밖의 적절한 에러 코드로 응답한다. 일치하는 Subject ID 와 타입을 포함하는 크리덴셜을 갖고 있는 것을 확인하면, Server 는 DTLS 프로토콜을 진행해서 Client 와 Server 가 결과적인 premaster secret 을 산출한다.

10.2 Raw 비 대칭 키 크리덴셜을 사용한 Device 인증

인증에 raw 비 대칭 키를 사용할 때, Client 와 Server 는 Device 에 바인딩된 크리덴셜로부터의 적절한 공개 키를 포함한다. 각 Device 는 제공된 공개 키가 갖고 있는 크리덴셜의 PublicData 필드와 일치하는지 확인하고, 크리덴셜의 대응하는 Subject ID 를 사용해서 피어 Device 를 식별한다.

10.3 Certificate 를 사용한 Device 인증

인증에 인증서를 사용할 때, Client 와 Server 는 각각 선택된 인증 cipher suite 의 일환으로 적절한 크리덴셜에 저장된 인증서 체인을 포함한다. 각 Device 는 피어 Device 에 의해 주어진 인증서 체인을 검증한다. 각각의 인증서 서명은 `oic.sec.cred.trustca' credusage 를 사용해서 /oic/sec/cred Resource 내에서 공개 키를 찾을 때까지 검증된다. /oic/sec/cred 에서 찾은 크리덴셜 Resource 는 인증서 경로 검증을 종료하는데 사용된다. 또한, 위의 모든 인증서에 대해 유효 기간 및 폐기 상태도 확인된다.

Device 는 of RFC 5280 의 Section 6 에 기술된 경로 검증 알고리즘을 따라야 한다. 특히,

- 모든 비 종단 개체 인증서에 대해, Device 는 기본 제약 확장자가 존재하고 확장자 내의 CA boolean 이 TRUE 인지 확인한다. 둘 중에 하나가 거짓이면 인증서 체인을 거절해야 한다. pathLenConstraint 필드가 존재하면, Device 는 이 인증서와 종단 개체 인증서 간의 인증서 수가 pathLenConstraint 이하인지 확인한다. 특히, pathLenConstraint 가 영이면, 이 인증서에 의해 종단 개체 인증서만 발행할 수 있다. pathLenConstraint 필드가 존재하지 않으면 체인 길이에 제한이 없다.

- 모든 비 종단 개체 인증서에 대해, Device 는 키 용도 확장자가 존재하고 keyCertSign 비트가 표명되어 있는지 확인한다.

- Device 는 Authority Key Identifier 확장자를 사용해서 발행 인증서를 신속하게 찾을 수 있다. Device 는 이 확장자가 없다고 해서 인증서를 거절할 수 없으며, 대신에 주체 명칭이 이 인증서의 발행자 명칭과 동일한 가능한 발행자 인증서의 공개 키를 사용해서 인증을 시도해야 한다.

- 체인의 종단 개체 인증서가 제공되는 목적에 적합한 Extended Key Usage (EKU)를 포함하는지를 확인한다. EKU 확장자가 없는 종단 개체 인증서는 어떠한 목적으로도 유효하지 않으며, 따라서 거절되어야 한다. anyExtendedKeyUsage OID (2.5.29.37.0)를 포함하는 인증서는 다른 유효한 EKU 를 갖고 있더라도 거절해야 한다.

- Device 는 특정 체인에 대해 "이행적 EKU"를 검증해야 한다. 체인 내의 발행자 인증서 (종단 개체가 아닌 모든 인증서)는 인증서 체인이 제공되는 목적에 대해 모두 유효해야 한다. EKU 확장자를 포함하고 해당하는 목적에 대해 EKU OID 가 확장자 내에 포함되어 있거나 EKU 확장자를 포함하지 않으면 발행자 인증서는 그 목적에 대해 유효하다. 발행자 인증서는 EKU 확장자 및 인증서 발행 인가 목적을 위한 EKU 의 완전한 목록을 포함해야 한다. EKU 확장자가 없는 발행자 인증서는 모든 목적에 대해 유효하며, 이는 EKU 확장자가 없는 종단 개체 인증서와 다른 점이다.

목적 목록 및 관련된 OID 는 섹션 9.3.2.2 에 정의된다.

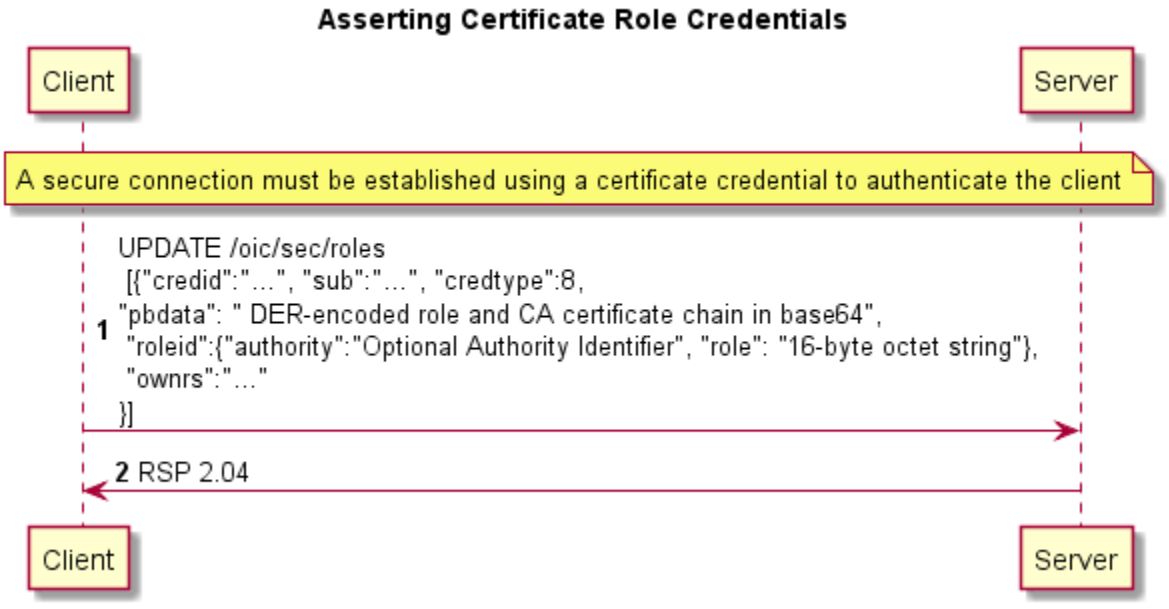
Device 가 확장자를 인식하지 못하면 크리티컬 필드를 조사해야 한다. 이 필드가 TRUE 이면, Device 는 인증서를 거절해야 한다. 이 필드가 FALSE 이면, Device 는 확장자가 없는 것처럼 인증서를 취급해서 처리를 진행해야 한다. 이것은 체인 내의 모든 인증서에 적용된다.

주: 인증서 폐기 메커니즘은 본 스펙의 현재 버전의 범위에 포함되지 않는다.

10.3.1 인증서를 사용한 Role 행사

이 섹션에서는 인증서 role 크리덴셜을 사용한 server 에 대한 client 의 role 주장에 대해 설명한다. server 가 인증서 크리덴셜 타입을 지원하지 않으면, client 는 인증서를 사용해서 role 을 주장해서는 안된다.

인증서를 통한 인증에 이어서, client 는 사용하고자 하는 role 인증서를 사용해서 server 의 role resource 를 갱신함으로써 하나 이상의 role 을 주장할 수 있다. role 크리덴셜은 인증서 크리덴셜이어야 하며, 인증서 체인을 포함한다. server 는 섹션 10.3 에 기술된 바와 같이 각각의 크리덴셜 체인을 검증한다. 또한, Device 인증에 사용된 종단 개체 인증서의 공개 키는 모든 role (종단 개체) 인증서의 공개 키와 일치해야 한다. 뿐만 아니라, 종단 개체 인증의 주체 구별 명칭과 role 인증서는 서로 일치해야 한다. 주장된 role 은 인증서의 subjectAltName 확장자 내에 인코딩된다. subjectAltName 필드는 복수의 값을 가질 수 있어서, 단일 인증서로 client 에 적용되는 복수의 role 을 인코딩할 수 있다. server 는 role 인증서의 EKU 확장자가 값 1.3.6.1.4.1.44924.1.7 (섹션 9.3.2.1 참조)을 포함해서 인증서가 role 을 주장하는데 사용될 수 있는지도 확인한다. 도 34 는 client Device 가 server 에 대해 role 을 주장하는 방법을 보여준다.



도 34 – 인증서 role 크리덴셜을 사용한 role 행사

도 34 주

1. 응답은 성공을 나타내는 "204 No Content" 또는 에러를 나타내는 4xx 를 포함한다. server 가 인증서 크리덴셜을 지원하지 않으면 "501 Not Implemented"를 리턴해야 한다.

2. client 에 의해 주장된 role 은 server 에 의해 선택된 기간 동안 유지될 수 있다. 이 기간은 role 인증서의 유효 기간을 초과할 수 없다. 새로운 CRL 정보가 획득되면, /oic/sec/roles 내의 인증서를 확인해서, 인증서가 폐기 또는 만료되었으면 role 을 제거해야 한다.

   server 는 client 에 의한 빈번한 role 의 재 주장을 피하기 위해 영이 아닌 기간을 선택해야 한다. 인증서의 유효 기간을 이 기간으로 사용함으로써 CMS 가 이 기간을 효과적으로 정할 수 있도록 하는 것이 바람직하다.

3. 생성 호출에서 전송되는 데이터의 형식은 크리덴셜의 목록이다 (oic.sec.cred, 표 23 참조). 데이터에는 credtype 8 (인증서를 가리킴)이 포함되고 PrivateData 필드는 존재하지 않는다. oic.sec.cred 객체와 인증서에 중복되는 필드에 대해서는 인증서의 값이 검증에 사용된다. 예를 들어, 크리덴셜에 Period 필드가 설정되어 있으면 server 는 인증서의 유효 기간을 신뢰할 수 있는 것으로 취급해야 한다. roleid 데이터 (authority, role)에 대해서도 마찬가지로 적용된다.

4. 인증서는 도 31 과 같이 인코딩된다 (base64 로 DER 인코딩된 인증서 체인).

5. Client 는 /oic/sec/roles resource 를 취득해서 이전에 주장했던 role 을 파악할 수 있다. 크리덴셜 객체의 배열이 리턴되고, 그렇지 않으면 현재 유효한 이전에 주장된 role 이 없음을 나타내는 "204 No Content"가 리턴된다.

11 메시지 무결성 및 기밀성

Client 와 Server 간의 보안 통신은 메시지 기밀성 및 무결성을 제공하는 보안 메커니즘을 사용해서 도청, 정보 조작, 또는 메시지 재생으로부터 보호된다.

11.1 DTLS 를 사용한 세션 보호

Device 는 [RFC 6347]에 정의된 보안 통신을 위한 DTLS 를 지원한다. TCP 를 사용하는 Device 는 [RFC 5246]에 정의된 보안 통신을 위한 TLS v1.2 를 지원한다. 메시지 통신을 위한 필수 또는 옵션 cipher suite 의 목록은 섹션 11.2 를 참조하기 바란다.

OCF Device 는 (D)TLS 버전 1.2 이상을 지원해야 하며, 버전 1.1 이하는 지원하지 않아야 한다.

주: 멀티캐스트 세션 의미는 본 버전의 security 스펙에는 정의되지 않는다.

11.1.1 유니캐스트 세션 의미

Client 와 Server 간의 유니캐스트 메시지에 대해서는 양쪽의 Device 가 상호 인증한다. Device 인증에 관한 자세한 사항은 섹션 10 을 참조하기 바란다.

Client 와 Server 간의 보안 유니캐스트 메시지에는 섹션 11.2 의 cipher suite 를 사용한다. 송신측 Device 는 선택한 cipher suite 에 정의된 대로 메시지를 암호화 및 인증하고, 수신측 Device 는 메시지를 처리하기 전에 메시지를 검증하고 복호화 해야 한다.

11.2 Cipher Suite

사용 허가된 cipher suite 는 컨텍스트에 따라 달라질 수 있다. 이 섹션에서는 소유권 이전과 통상적인 작동 중에 허가되는 cipher suite 를 나열한다. 다음의 RFC 는 OCF 에서 사용되는 cipher suite 에 관한 추가 정보를 제공한다.

[RFC 4279]: (D)TLS 에서 사전 공유 키 (PSK)의 사용을 규정한다.
[RFC 4492]: (D)TLS 에서 타원 곡선 암호 기법의 사용을 규정한다.
[RFC 5489]: 타원 곡선 Diffie-Hellman (ECDHE) 및 PSK 를 사용하는 cipher suite 의 사용을 규정한다.
[RFC 6655, 7251]: ECDHE 와 함께 AES-CCM 모드 cipher suite 를 규정한다.

11.2.1 Device 소유권 이전을 위한 Cipher Suite

11.2.1.1 Just Works 방식 Cipher Suite

Just Works 소유권 이전 방법은 다음의 (D)TLS cipher suite 를 사용할 수 있다.

TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256,
TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256

Just Works OTM 을 지원하는 모든 Device 는 다음을 구현한다.

TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256 (값 0xFF00 사용)

Just Works OTM 을 지원하는 모든 Device 는 다음을 구현해야 한다.

TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256 (값 0xFF01 사용)

11.2.1.2 Random PIN 방식 Cipher Suite

Random PIN 기반 소유권 이전 방법은 다음의 (D)TLS cipher suite 를 사용할 수 있다.

TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

Random Pin 기반 OTM 을 지원하는 모든 Device 는 다음을 구현한다.

TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256

11.2.1.3 인증서 방식 Cipher suite

제조사 인증서 기반 소유권 이전 방법은 다음의 (D)TLS cipher suite 를 사용할 수 있다.

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

다음의 커브를 사용해서

secp256r1 ([RFC4492] 참조)

제조사 인증서 기반 OTM 을 지원하는 모든 Device 는 다음을 구현한다.

TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

제조사 인증서 기반 OTM 을 지원하는 Device 는 다음을 구현해야 한다.

TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

TLS_ECDHE_ECDSA_WITH_AES_256_CCM

11.2.2 대칭 키용 Cipher Suite

다음의 cipher suite 는 PSK 를 사용하는 (D)TLS 통신에 대해 정의된다.

TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

TLS_PSK_WITH_AES_128_CCM_8, (* 8 OCTET 인증 태그 *)

TLS_PSK_WITH_AES_256_CCM_8,

TLS_PSK_WITH_AES_128_CCM, (* 16 OCTET 인증 태그 *)

TLS_PSK_WITH_AES_256_CCM,

주: 모든 CCM 기반 cipher suite 는 인증을 위해 HMAC-SHA-256 도 사용한다.

모든 Device 는 다음을 구현한다.

TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

Device 는 다음을 구현해야 한다.

TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_PSK_WITH_AES_256_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

11.2.3 비 대칭 크리덴셜용 Cipher Suite

다음의 cipher suite 는 비 대칭 키 또는 인증서를 사용하는 (D)TLS 통신에 대해 정의된다.

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

다음의 커브를 사용하여

secp256r1 (See [RFC4492])

비 대칭 크리덴셜을 지원하는 모든 Device 는 다음을 구현한다.

TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

비 대칭 크리덴셜을 지원하는 모든 Device 는 다음을 구현해야 한다.

TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,
TLS_ECDHE_ECDSA_WITH_AES_128_CCM,
TLS_ECDHE_ECDSA_WITH_AES_256_CCM

12 액세스 제어

12.1 ACL 생성 및 관리

이 섹션은 향후 버전의 스펙에서 확장될 예정이다.

12.2 ACL 평가 및 시행

Server 는 요청자에게 노출시키기 전에 애플리케이션 Resource 에 대해 액세스 제어를 시행한다. 보안 포트를 통해 액세스를 수신하면, Server 내의 Security Resource Manager (SRM)는 요청자를 인증한다. "주체"로 불리는 인증된 요청자는 요청자의 아이덴티티, role 을 특정하는 ACL 항목의 매칭에 사용될 수 있다. 또는, 주체 와일드카드를 사용해서 인증된 요청자를 매칭할 수 있다.

비 보안 포트를 통해 요청이 수신되면, 익명의 요청자의 주체 와일드카드 매칭을 사용하는 ACL 정책만 허용된다.

요청된 resource 가 ACL 항목에 매칭되지 않으면 액세스가 거절된다. (주: 보안 가상 resource 에의 액세스가 ACL resource 의 프로비저닝에 앞서서 허용되는 경우가 있는 Device 온보딩에 관련된 문서화된 예외가 있다.)

2 세대 ACL (즉, /oic/sec/acl2)은 resource reference 의 배열을 사용해서 ACE2 액세스 정책이 적용되는 Resource 를 매칭하는 resource 매칭 알고리즘을 사용하는 Access Control Entries (ACE2)의 배열을 포함한다. 매칭은 ACE2 "resource" property 의 값 (섹션 13 참조)와 요청된 Resource 의 비교를 통해 이루어진다. Resource 는 다음과 같은 네 가지 방법 중 하나를 사용해서 매칭된다: host reference (href), resource type (rt), resource interface (if), 및 resource wildcard.

12.2.1 Host Reference 매칭

ACE2 매칭 요소 내에 존재하는 경우, Host Reference (href) Property 가 resource 매칭에 사용된다.

- href Property 는 Resource 명칭의 정확한 매칭을 찾기 위해 사용된다.

12.2.2 Resource Type 매칭

ACE2 매칭 요소 내에 존재하는 경우, Resource Type (rt) Property 가 resource 매칭에 사용된다.

- rt Property 는 Resource Type 명칭의 정확한 매칭을 찾기 위해 사용된다.
- 복수의 Resource Type 명칭을 구현하는 Resource (예: collection resource)의 매칭에는 스트링의 배열이 사용된다.

12.2.3 Interface 매칭

ACE2 매칭 요소 내에 존재하는 경우, Interface (if) property 가 resource 매칭에 사용된다.

- 'if' property Resource Interface 스트링의 정확한 매칭을 찾기 위해 사용된다.
- Resource 가 복수의 Interface 를 구현하는 경우에는 스트링의 배열이 사용된다.

12.2.4 다중 기준 매칭

동일한 ACE2 Resource property 내에 복수의 매칭 기준이 제공되면 (예: 'href' 및 'rt' 및 'if') 기준의 논리 AND 가 적용된다. 예를 들어, Resource property 내에 'href'="/a/light 와 'if'="oic.if.s"가 있으면 후보 resource 에 대해 'href'와 'if' 기준이 둘 다 참일 때 매칭이 존재하게 된다.

ACE2 "resource" property 가 항목의 배열이면, 각 배열 요소에 대해 논리 OR 가 적용된다. 예를 들어, Resource property 의 첫 번째 배열 요소가 'href'="/a/light"를 포함하고 Resource property 의 두 번째 배열 요소가 'if'="oic.if.s"를 포함하면 'href' 기준 또는 'if'기준에 매칭되는 Resource 가 매칭 Resource 집합에 포함된다.

12.2.5 Resource 와일드카드 매칭

와일드카드 표현을 사용함으로써 oic.sec.ace2.resource-ref 구조에 포함된 와일드카드 Property 를 사용해서 복수의 Resource 를 매칭할 수 있다. 다음과 같은 와일드카드 매칭 스트링이 정의된다.

스트링	설명
"+"	모든 발견 가능 resource 에 매칭된다.
"-"	모든 발견 불가능 resource 에 매칭된다.
"*"	모든 resource 에 매칭된다.

표 18 – ACE2 와일드카드 매칭 스트링 설명

주: 발견 불가능 resource 는 다른 collection resource 에 나타날 수 있지만 /res collection 에는 나타나지 않는 반면에, 발견 가능 resource 는 /oic/wk/res Resource 에 나타난다.

예: Resource 매칭을 위한 JSON

```
{
  [
```



```
//Matches Resources named "/x/door1" or "/x/door2"
    {
      "href":"/x/door1"
    },
    {
      "href":"/x/door2"
    },
//Matches Resources with Resource Type "oic.sec.crl" and "oic.sec.cred"
    {
      "rt":[" oic.sec.crl ", "oic.sec.cred "]
    },
// Matches Resources that implement both "oic.if.baseline" and
"oic.if.rw" Interfaces.

          "if":["oic.if.baseline", "oic.if.rw"]
    },
//Matches Resources named "/x/light1" or "/x/light2" and have Resource Types "x.light.led",
"x.light.flourescent" and "x.light.color".
    {
      "href":"/x/light1",
          "rt":["x.light.led","x.light.flourescent", "x.light.color"]
    },
    {
      "href":"/x/light2",
          "rt":["x.light.led","x.light.flourescent", "x.light.color"]
    },
//Matches all Resources.
    {
      "wc":"*"
    }
  ]
}
```

12.2.6 와일드카드를 사용한 주체 매칭

ACE 주체가 와일드카드 스트링 "*"으로 지정되어 있으면 모든 요청자가 매칭된다. OCF server 는 OCF client 를 인증할 수 있지만 인증하도록 요구되는 것은 아니다.

예: 주체 와일드카드 매칭을 위한 JSON


```
//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"
}
```

12.2.7 Role 을 사용한 주체 매칭

ACE 주체가 role 로 지정되어 있으면, 다음과 같은 경우에 요청자가 매칭된다.

1. 요청자가 대칭 키 크리덴셜로 인증되고 크리덴셜 resource 의 크리덴셜 항목의 roleid property 내에 role 이 존재하거나,
2. 요청자가 인증서로 인증되고 평가 시에 요청자의 인증서 공개 키와 함께 role resource 내에 유효한 role 인증서가 존재할 때. role 인증서의 검증은 섹션 10.3.1 에 정의된다.

12.2.8 ACL 평가

OCF Server 는 다음과 같은 시퀀스로 매칭을 수행하는 ACE2 매칭 알고리즘을 적용한다.

1. /oic/sec/sacl Resource 가 존재하고 서명이 성공적으로 검증되면 이들 ACE2 항목이 3 단계에서 로컬 ACE2 항목의 집합에 기여한다. Server 는 /oic/sec/sacl Resource 의 갱신에 이어서 최소한 한번 서명을 검증한다.
2. 로컬 /oic/sec/acl2 Resource 는 자신의 ACE2 항목을 매칭에 제공한다.
3. 다음의 모든 기준이 만족되면 액세스가 허가된다.
    a. 요청자가 ACE2 "주체" Property 에 의해 매칭된다.
    b. 요청된 Resource 가 ACE2 "resource" Property 에 의해 매칭되고 요청된 Resource 가 로컬 Server 상에 존재한다.
    c. "period" Property 제약이 만족된다.
    d. "permission" Property 제약이 적용된다.

주: 복수의 ACE2 항목이 Resource 요청에 매칭되면, 모든 매칭 ACE 의 통합으로 *유효* 허가를 정의한다. 예를 들어, Perm1=CR---; Perm2=--UDN 이면, UNION (Perm1, Perm2)=CRUDN.

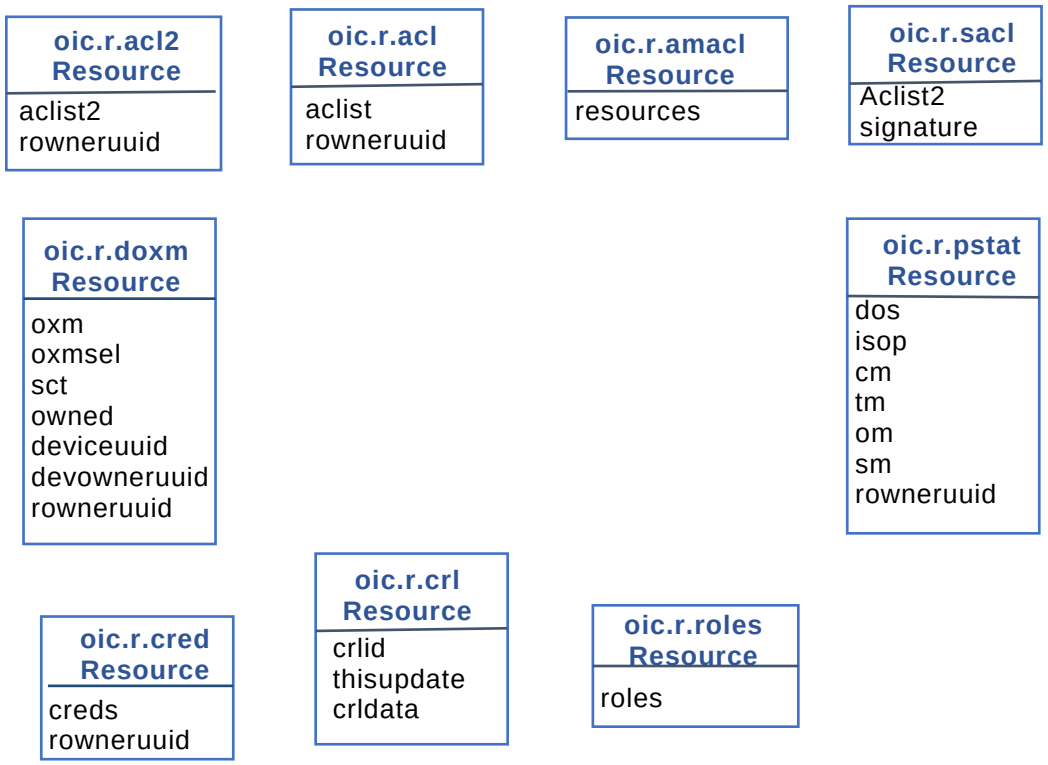
2689

2690 Server 는 유효 허가를 토대로 액세스를 시행한다.

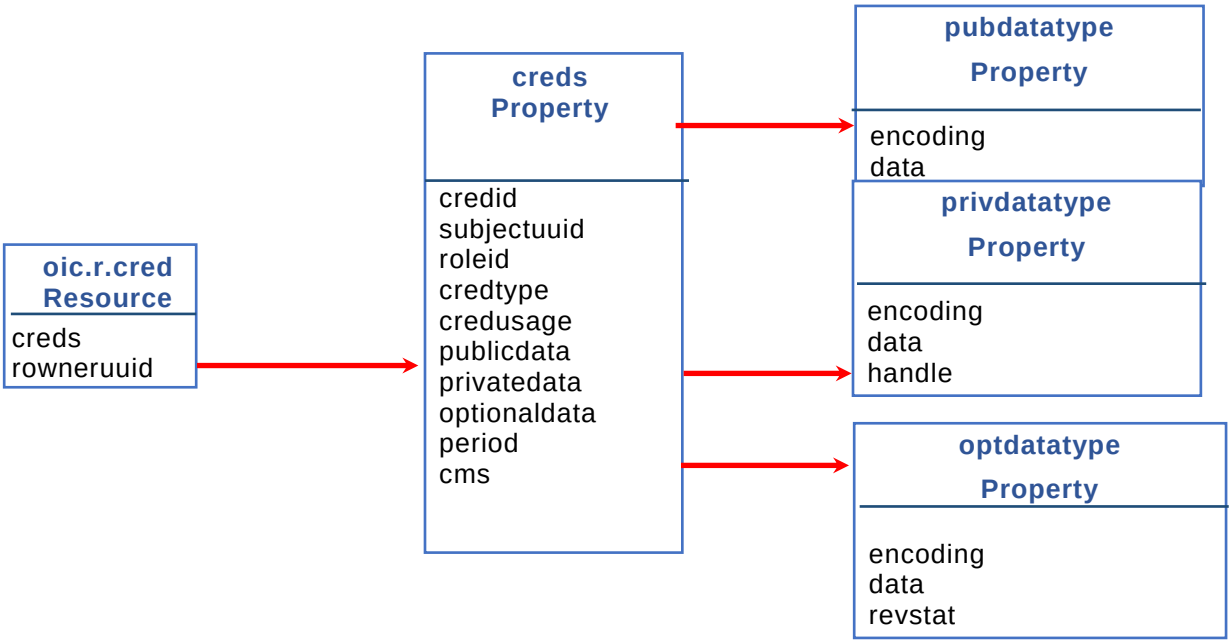
2691

2692

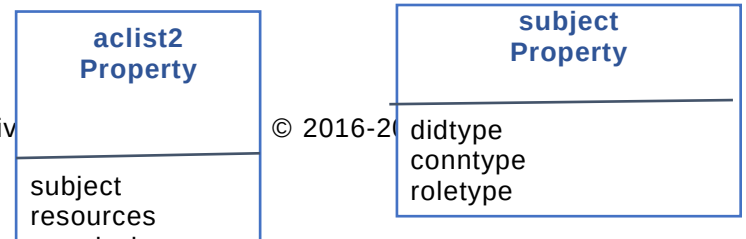
2693 **13 보안 Resource**

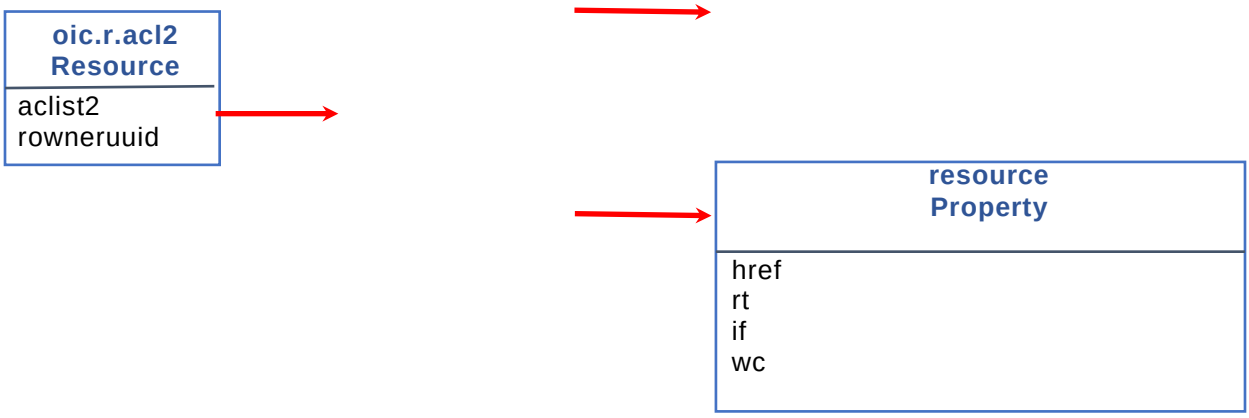


2694 **도 35 – OCF 보안 Resource**

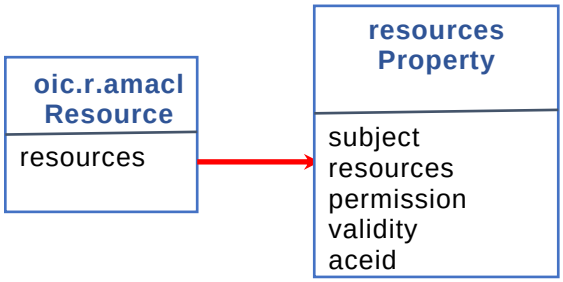


2695 **도 36 – oic.r.cred Resource 및 Property**

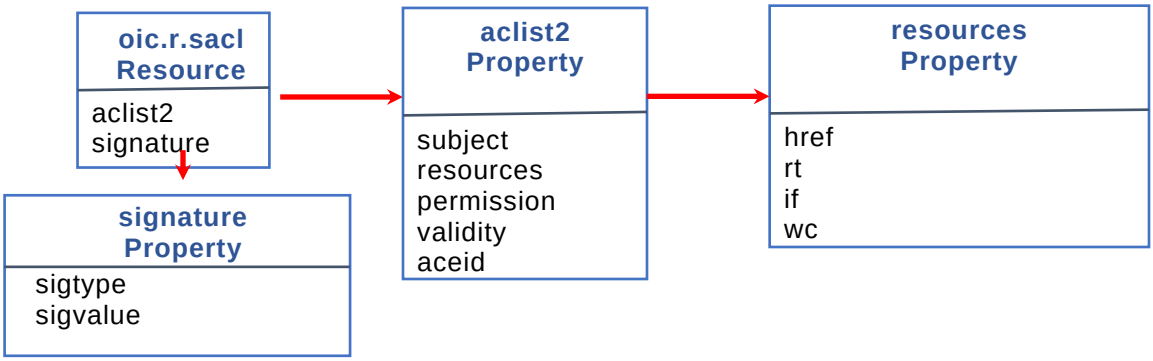




2696 **도 37 – oic.r.acl2 Resource 및 Property**



2697 **도 38 – oic.r.amacl Resource 및 Property**



2698 **도 39 – oic.secr.sacl Resource 및 Property**

2699 **13.1 Device Owner Transfer Resource**

2700 /oic/sec/doxm Resource 는 일련의 지원되는 Device 소유권 이전 방법을 포함한다.

2701 Resource discovery 프로세스는 본 스펙에 포함된 보안 Resource 정의의 일환으로 제공되는
2702 CRUDN 제약을 준수한다.

고정 URI	Resource Type Title	Resource Type ID ("rt" 값)	Interface	설명	관련 기능 연동
/oic/sec/doxm	Device Owner Transfer Methods	urn:oic.r.doxm	baseline	Device 소유권 이전을 지원하기 위한 Resource	구성

2703 **표 19 – oic.r.doxm Resource 정의**

Property Title	Property Name	Value Type	Value Rule	필수	Device 상태	액세스 모드	설명
Owner Transfer Method	oxms	oic.sec.doxmtype	배열	예		R	소유권 이전 방법을 식별하기 위한 값 및 방법을 정의한 기구
Oxm Selection	oxmsel	oic.sec.doxmtype	UINT16	예	RESET	R	Server 는 (4) "oic.sec.oxm.self"로 설정한다.
					RFOTM	RW	아직 인증되지 않은 DOXS 는 선택한 OTM 으로 설정하고 양쪽이 OTM 을 실행한다. 보안 소유권 이전 세션이 구성되고 나면, DOXS 는 oxmsel 을 다시 갱신해서 영구적으로 만든다. OTM 이 실패하면 Server 는 Device 상태를 RESET 으로 전환한다.
					RFPRO	R	해당 없음
					RFNOP	R	해당 없음
					SRESET	R	해당 없음
Supported Credential Types	sct	oic.sec.credtype	bitmask	예		R	Device 가 지원하는 크리덴셜의 타입을 식별한다. SRM 은 보안 기능을 결정한 후 framework 초기화 시에 이 값을 설정한다.

Owned	owned	Boolean	T|F	예	RESET	R	Server 는 FALSE 로 설정한다.
					RFOTM	RW	보안 소유권 이전 세션이 구성되고 나면 DOXS 는 TRUE 로 설정한다.
					RFPRO	R	해당 없음
					RFNOP	R	해당 없음
					SRESET	R	해당 없음
Device UUID	deviceuuid	스트링	oic.sec.didtype	예	RESET	R	Server 는 RESET 으로 전환할 때마다 다른 임시 random UUID 를 구성한다.
					RFOTM	RW	보안 소유권 이전 세션이 구성되고 나면 DOXS 가 선택한 값으로 갱신한다. 에러 PROPERTY_NOT_FOUND 로 갱신을 실패하면, DOXS 는 Server 가 제공하는 값을 받아들이거나 /doxm.owned=FALSE 로 갱신하고 세션을 종료한다.
					RFPRO	R	해당 없음
					RFNOP	R	해당 없음
					SRESET	R	해당 없음
Device Owner Id	devowneruuid	스트링	uuid	예	RESET	R	Server 는 nil uuid 값으로 설정한다 (예: "00000000-0000-0000-0000-000000000000")
					RFOTM	RW	보안 소유권 이전 세션이 구성되고 나면 DOXS 가 값을 설정한다.
					RFPRO	R	해당 없음

					RFNOP	R	해당 없음
					SRESET	R	해당 없음
Resource Owner Id	rowneruuid	스트링	uuid	예	RESET	R	Server 는 nil uuid 값으로 설정한다 (예: "00000000-0000-0000-0000-000000000000")
					RFOTM	RW	소유권 이전 세션이 성공적으로 구성되면 DOXS 는 rowneruuid property 를 구성해야 한다.
					RFPRO	R	해당 없음
					RFNOP	R	해당 없음
					SRESET	RW	DOXS (/doxm.devowneruuid property 를 통해 참조)는 상호 인증된 보안 세션이 구성되면 resource owner property 를 검증하고, 필요하면, 갱신해야 한다. Rowneruuid 가 유효한 DOXS 를 참조하지 않으면 Server 는 RESET Device 상태로 전환한다.

2704 **표 20 – oic.r.doxm Resource 의 Property**

Property Title	Property Name	Value Type	Value Rule	필수	Device 상태	액세스 모드	설명
Device ID	uuid	스트링	uuid	예	RW	-	uuid 값

2705 **표 21 - oic.sec.didtype Property 의 속성**

2706 소유권 이전 방법 (oxms) Property 는 선호 순서대로 나열된 소유권 이전 방법의 목록을 포함한다.
2707 Device 제조사는 가장 바람직한 방법이 우선도가 낮은 방법의 앞에 오도록 이 Property 를 구성한다.

네트워크 관리 툴이 가장 적합한 방법을 선택할 때, 온보딩 시에 네트워크 관리 툴이 이 목록을 검색한다.

소유권 이전 방법의 선택에 이어서, oxmsel Property 를 사용해서 합의된 방법이 /doxm Resource 에 입력된다.

소유권 이전 방법은 공급자 또는 기구를 식별하기 위한 URN 및 특정 방법의 두 부분으로 구성된다.

<OxmType> ::= "urn:" <NID> ":" <NSS>

<NID> :: = <Vendor-Organization>

<NSS> ::= <Method> | {<NameSpaceQualifier> "."} <Method>

<NameSpaceQualifier> ::= String

<Method> ::= String

<Vendor-Organization> ::= String

소유권 이전 방법이 성공적으로 완료되면, *owned* Property 가 '1' (TRUE)로 설정된다. 그 결과, 해당하는 Device 의 소유권을 취득하기 위한 이후의 시도는 실패하게 된다..

성공적인 소유권 이전에 응하여 SRM 은 /oic/sec/doxm Resource 에 저장되는 Device identifier (deviceuuid)를 생성한다.

소유권 이전 방법은 소유권을 취득하는 서비스에 deviceuuid 를 전달해야 한다. 서비스는 보안 데이터베이스 내에 deviceuuid 와 OC 를 관련 지어야 한다.

Device 공급자는 Device identifier (deviceuuid)가 지속적이거나 (갱신 불가) 비 지속적인 것으로 (소유권 이전 서비스에 의해 갱신 가능 – a.k.a DOXS) 판단한다.

deviceuuid 가 지속적인 것이면, 갱신 요청은 에러 PROPERTY_NOT_FOUND 로 실패하게 된다.

비 지속적이면, 갱신 요청이 성공적으로 수행되고 Device 가 RESET 될 때까지 DOXS 에 의해 제공된 값이 기억된다. 그 밖의 다른 이유로 갱신이 실패하고 Device 상태가 RESET 으로 전환되지 않으면, deviceuuid 의 값은 nil UUID (예: "00000000-0000-0000-0000-000000000000")로 된다.

2735 Device 의 deviceuuid 가 지속성에 관계 없이, Device 가 RESET 상태로 전환될 때마다 임시 random
2736 비 반복 UUID 가 생성된다. 임시 deviceuuid 는 DOXS 가 보안 OTM 연결을 구성할 때까지 Device
2737 상태가 RESET 상태인 동안 그리고 RFOTM Device 상태인 동안 사용된다.

2738

13.1.1 OCF 정의 소유권 이전 방법

Value Type Name	Value Type URN (옵션)	열거 값 (필수)	설명
OCFJustWorks	oic.sec.doxm.jw	0	just-works 방법은 anonymous Diffie-Hellman 키 합의 프로토콜에 의존해서 OBT 가 신규 Device 의 소유권을 주장할 수 있도록 한다. 첫 번째로 주장하는 OBT 가 Device 소유자로 인정된다. just-works 방법은 Device to the OBT 에 대해 Device 를 인증하고 마찬가지로 Device 에 대해 OBT 를 인증하는데 사용되는 공유 비밀을 생성한다. Device 는 OBT 가 Device 의 소유권을 취득하도록 하고, 다른 OBT 에 의한 이후의 소유권 취득 시도는 실패하게 된다. 주: just-works 방법은 중간 공격자의 공격 대상이 된다. 따라서, 이 방법을 사용할 때는 물리적 보안을 제공하기 위한 사전 대책이 있어야 한다.
OCFSharedPin	oic.sec.doxm.rdp	1	신규 Device 가 대역 외 채널을 통해 Device OBT 로 전달되는 PIN 을 무작위로 생성한다. 대역 내 Diffie-Hellman 키 합의 프로토콜은 양쪽 종단 점이 PIN 을 소유하도록 구성한다. OBT 에 의한 PIN 의 소유는 신규 Device 에 device 소유권을 주장할 수 있음을 시사한다.
OCFMfgCert	oic.sec. doxm.mfgcert	2	신규 Device 는 Device 온보딩 시에 Diffie-Hellman. 교환에 서명하는데 사용되는 비 대칭 개인 키를 내장해서 제조되는 것으로 간주한다. 제조사 인증서는 Platform 강화 정보 및 그 밖의 보안 보증 주장을 포함해야 한다.
OCF Reserved	<Reserved>	3	예약
OCFSelf	oic.sec.oxm.self	4	제조사는 /doxm.oxmsel 값을 (4)로 설정한다. Server 는 RESET Device 상태로 전환 시에 이 값을 (4)로 리셋한다.
OCF Reserved	<Reserved>	5~0xFEFF	OCF 사용을 위해 예약
Vendor-defined Value Type Name	<Reserved>	0xFF00~0xFFFF	공급자 특정 OTM 사용을 위해 예약

표 22 – oic.sec.doxmtype Property 의 속성

13.2 크리덴셜 Resource

/oic/sec/cred Resource 는 Client 를 검증하고 서비스를 지원하는데 사용되는 크리덴셜뿐만 아니라 Client 에 대해 Server 를 인증하고 서비스를 지원하는데 사용되는 크리덴셜도 보유한다.

OCF framework 에서는 쌍 사전 공유 키, 비 대칭 키, 인증서 등을 포함하는 복수의 크리덴셜 타입이 허용된다. credential Resource 는 Subject UUID 를 사용해서 인증 시도를 검증함으로써 Client 를 식별하고 인식한 서비스를 지원한다.

"creds" Array Property 를 관리하기 위한 interface 를 제공하기 위해, oic.r.cred Resource 상의 RETRIEVE, UPDATE, 및 DELETE 동작은 다음과 같이 이루어진다.

1. RETRIEVE 는 쓰기 전용 Property (예: 개인 키 데이터)를 생략하는 것을 제외하고 전체 Resource 표현을 리턴한다.
2. UPDATE 는 다음과 같이 UPDATE 요청과 함께 전송된 표현에 포함된 Property 를 대체하거나 Property 에 추가한다.
    a. UPDATE 표현이 "creds" array Property 를 포함하면,
        i. 기존의 "credid"와 일치하는 "credid"로 제공된 creds 는 기존의 "creds" 배열 내의 대응하는 cred 를 완전히 대체한다.
        ii. "credid" 없이 제공된 creds 는 기존의 "creds" 배열에 추가되고, Server 에 의해 고유한 (cred Resource 에 대해) "credid"가 생성되어 신규 cred 에 할당된다. interface 의 결정성을 향상시키고 경합 조건을 완화시키기 위해 삭제된 cred 의 "credid"는 재 사용하지 않는다.
        iii. 기존의 "credid"와 일치하지 않는 "credid"로 제공된 creds 는 제공된 "credid"를 사용해서 기존의 "creds" 배열에 추가된다.
3. 질의 파라미터가 없는 DELETE 는 전체 "creds" 배열을 삭제하지만 oic.r.cred Resource 는 삭제하지 않는다.
4. 하나 이상의 "credid" 질의 파라미터를 갖는 DELETE 는 "creds" 배열로부터 대응하는 credid 를 갖는 cred 를 삭제한다.

고정 URI	Resource Type Title	Resource Type ID ("rt" 값)	Interface	설명	관련 기능 연동
/oic/sec/cred	Credentials	urn:oic.r.cred	baseline	Device 인증, 검증, 및 데이터 보호를 위한 크리덴셜을 포함하는 Resource.	보안

2769 **표 23 –oic.r.cred Resource 정의**

Property Title	Property Name	Value Type	Value Rule	필수	Device 상태	액세스 모드	설명
Credentials	creds	oic.sec.cred	배열	예	RESET	R	Server 는 제조사 기본설정으로 설정한다.
					RFOTM	RW	성공적인 OTM 후에 DOXS 에 의해 설정된다.
					RFPRO	R	성공적인 인증 후에 CMS (/cred.rowneruuid property 를 통해 참조)에 의해 설정된다. vertical resource 에 대한 액세스는 금지된다.
					RFNOP	R	매칭 ACE 를 찾으면 vertical resource 에 대한 액세스가 허용된다.
					SRESET	RW	보안 세션이 구성되고 Server 와 DOXS 가 인증되면 DOXS (/doxm.devowneruuid property 를 통해 참조)는 creds 항목의 무결성을 평가해야 하고 creds 항목을 갱신할 수 있다.
Resource Owner ID	rowneruuid	스트링	uuid	예	RESET	R	Server 는 nil uuid 값 (예: "00000000-0000-0000-0000-000000000000")으로 설정한다.
					RFOTM	RW	소유권 이전 세션이 성공적으로 구성되면 DOXS 는 /cred.owneruuid property 를 구성해야 한다.
					RFPRO	R	해당 없음

					RFNOP	R	해당 없음
					SRESET	RW	상호 인증된 보안 세션이 구성되면 DOXS (/doxm.devowneruuid property 를 통해 참조)는 resource owner property 를 검증하고, 필요하면 갱신해야 한다. rowneruuid 가 유효한 DOXS 를 참조하지 않으면 Server 는 RESET Device 상태로 전환한다.

표 24 –oic.r.cred Resource 의 Property

모든 보안 Device 액세스는 종단 간 상호 작용을 보호하는 /oic/sec/cred Resource 를 갖는다.

/oic/sec/cred Resource 는 ‘rowneruuid’’ Property 내에 명명된 서비스에 의해 생성 및 수정될 수 있다.

ACL 명명 /oic/sec/cred Resource 는 CRUDN 액세스 모드를 초과하는 액세스를 더 제한해야 한다.

Property Title	Property Name	Value Type	Value Rule	필수	액세스 모드	Device 상태	설명
Credential ID	credid	UINT16	0 – 64K-1	예	RW		다른 Resource 로부터의 로컬 참조를 위한 짧은 크리덴셜 ID.
Subject UUID	subjectuuid	스트링	uuid	예	RW		이 크리덴셜이 적용되는 주체를 식별하기 위한 uuid.
Role ID	roleid	oic.sec.roletype	-	아니오	RW		주체가 주장하도록 인가된 role 을 식별한다.
Credential Type	credtype	oic.sec.credtype	bitmask	예	RW		이 크리덴셜 타입을 표현한다. 0 – 검사에 사용 1 – 대칭 쌍 키 2 – 대칭 그룹 키 4 – 비 대칭 서명 키 8 – 인증서 달린 비 대칭 서명 키 16 – PIN 또는 비밀번호 32 – 비 대칭 암호 키
Credential Usage	credusage	스트링	-	아니오	RW		크리덴셜의 결정 불가능성을 해소하기 위해 사용된다. 어떻게/어디에서 cred 가 사용되는지를 가리킨다. oic.sec.cred.trustca: 인증서 신뢰 앵커 oic.sec.cred.cert: 아이덴티티 인증서 oic.sec.cred.rolecert: role 인증서 oic.sec.cred.mfgtrustca: 제조사 인증서 신뢰 앵커 oic.sec.cred.mfgcert: 제조사 인증서
Public Data	publicdata	oic.sec.pubdatatype	-	아니오	RW		공개 크리덴셜 정보 1:2: 티켓, 공개 SKDC 값 4, 32: 공개 키 값 8: 인증서

Private Data	privatedata	oic.sec.privdatatype	-	아니오			1:2: 대칭 키 4: 8, 32, 64: 개인 비 대칭 키 16: 비밀번호 해시, 비밀번호 값, 보안 질문
					-	RESET	Server 는 제조사 기본설정으로 설정한다.
					W	RFOTM	성공적인 OTM 후에 DOXS 에 의해 설정된다.
					W	RFPRO	인증된 DOXS 또는 CMS 에 의해 설정된다.
					-	RFNOP	정상 동작 중에는 쓸 수 없다.
					W	SRESET	DOXS 가 RFPRO 로의 전환이 가능하도록 수정할 수 있다.
Optional Data	optionaldata	oic.sec.optdatatype	-	아니오	RW		크리덴셜 폐기 상태 정보 1, 2, 4, 32: 폐기 상태 정보 8: 폐기 + CA 인증서
Period	period	스트링	-	아니오	RW		RFC5545 에 의해 정의된 기간. 현재 시각이 Period 시간대에서 벗어나면 크리덴셜을 사용하지 않는다.
Credential Refresh Method	crms	oic.sec.crmtype	배열	아니오	RW		Period Property 를 가진 크리덴셜은 oic.sec.crm 에 대한 타입 정의에 따라 크리덴셜 갱신 방법 (crm)을 사용해서 갱신된다.

2777 **표 25 –oic.sec.cred Property 의 속성**

Property Title	Property Name	Value Type	Value Rule	액세스 모드	필수	설명

Encoding format	encoding	스트링	-	RW	아니오	pubdata 에 포함된 데이터의 인코딩 형식을 지정하는 스트링 "oic.sec.encoding.jwt" - RFC7517 JSON 웹 토큰 (JWT) 인코딩 "oic.sec.encoding.cwt" - RFC CBOR 웹 토큰 (CWT) 인코딩 "oic.sec.encoding.base64" – Base64 인코딩 "oic.sec.encoding.uri" – URI 참조 "oic.sec.encoding.pem" –PEM 인코딩된 인증서 또는 체인에 대한 인코딩 "oic.sec.encoding.der" –DER-인코딩된 인증서 또는 체인에 대한 인코딩 "oic.sec.encoding.raw" – Raw hex 인코딩된 데이터
Data	data	스트링	-	RW	아니오	인코딩된 값.

2778 **표 26 –oic.sec.pubdatatype Property 의 속성**

Property Title	Property Name	Value Type	Value Rule	액세스 모드	필수	설명
Encoding format	encoding	스트링	-	RW	예	privdata 에 포함된 데이터의 인코딩 형식을 지정하는 스트링. "oic.sec.encoding.jwt" - RFC7517 JSON 웹 토큰 (JWT) 인코딩 "oic.sec.encoding.cwt" - RFC CBOR 웹 토큰 (CWT) 인코딩 "oic.sec.encoding.base64" – Base64 인코딩 "oic.sec.encoding.uri" – URI 참조 "oic.sec.encoding.handle" – 데이터가 핸들을 사용해서 참조되는 저장 서브 시스템 내에 포함된다. "oic.sec.encoding.raw" – Raw hex 인코딩된 데이터

Data	data	스트링	-	RW	아니오	인코딩 값 이 값은 절대 읽을 수 없다.
Handle	handle	UINT16	-	RW	아니오	키 저장 resource 에 대한 핸들

2779 **표 27 –oic.sec.privdatatype Property 의 속성**

Property Title	Property Name	Value Type	Value Rule	액세스 모드	필수	설명
Revocation status	revstat	Boolean	T | F	RW	예	폐기 상태 플래그 True – 폐기되었음 False – 폐기되지 않았음
Encoding format	encoding	스트링	-	RW	아니오	optdata 에 포함된 데이터의 인코딩 형식을 지정하는 스트링 "oic.sec.encoding.jwt" - RFC7517 JSON 웹 토큰 (JWT) 인코딩 "oic.sec.encoding.cwt" - RFC CBOR 웹 토큰 (CWT) 인코딩 "oic.sec.encoding.base64" – Base64 인코딩 "oic.sec.encoding.pem" –PEM 인코딩된 인증서 또는 체인을 위한 인코딩 "oic.sec.encoding.der" –DER 인코딩된 인증서 또는 체인을 위한 인코딩 "oic.sec.encoding.raw" – Raw hex 인코딩된 데이터
Data	data	스트링	-	RW	아니오	인코딩된 구조

2780 **표 28 –oic.sec.optdatatype Property 의 속성**

Property Title	Property Name	Value Type	Value Rule	액세스 모드	필수	설명

Authority	authority	스트링	-	R	아니오	role 을 정의한 기관의 명칭. 존재하지 않으면 크리덴셜 발행자가 role 을 정의한 것이다. 존재하면 ASN.1 PrintableString 으로 표현 가능해야 한다.
Role	role	스트링	-	R	예	role 을 식별하기 위한 식별자. ASN.1 PrintableString 으로 표현 가능해야 한다.

표 29 –oic.sec.roletype Property 정의

13.2.1 크리덴셜 Resource 의 Property

13.2.1.1 Credential ID

Credential ID (credid)는 /oic/sec/cred 인스턴스에 대한 로컬참조이다. credid 는 SRM 에 의해 생성된다. credid 는 동일한 Subject UUID 를 갖는 Resource 인스턴스들의 차이를 명확하게 하기위해 사용된다.

13.2.1.2 주체 UUID

Subject UUID 는 보안 세션을 구성하고, 인증 시도 응답을 검증하거나 인증 시도를 인증하기 위해 크리덴셜 Resource 가 사용되는 Device 또는 서비스를 식별한다.

Server 자신의 Device ID 와 일치하는 Subject UUID 는 해당하는 Device 를 인증하는 크리덴셜을 식별한다.

Subject UUID 는 공유 데이터를 보호하기 위해 그룹 키가 사용되는 그룹을 식별하는데 사용된다.

13.2.1.3 Role ID

Role ID 는 Subject UUID 에 부여된 role 의 집합을 식별한다. 주장된 role 또는 role 의 집합은 roleid Property 에 포함된 role 값의 부분집합이다.

크리덴셜이 role 의 집합을 포함하는 경우, 주장된 role 이 크리덴셜 내의 role 집합의 구성요소이면 ACL 이 성공적으로 매칭된다.

13.2.1.4 Credential Type

Credential Type 은 크리덴셜의 타입에 따라 컨텐츠가 달라질 수 있는 여러 다른 Property 값을 해석하는데 사용된다. 이러한 property 에는 publicdata, privatedata, 및 optionaldata 가 포함된다. CredType 값 '0' ("보안 모드 없음")은 시험 및 디버깅 환경을 위해 예약된다. 제품 배치는 타입

'0'인 크리덴셜의 프로비저닝을 허용하지 않는다. SRM 은 제품 배치에의 사용을 방지하기 위한 검사 코드를 도입해야 한다.

13.2.1.5 Public Data

Public Data 는 크리덴셜 발행을 둘러싼 추가적인 컨텍스트를 제공하는 정보를 포함한다. 예를 들어, 인증서 또는 Key Management Service 로부터의 응답 데이터 내에 포함된 정보를 포함할 수 있다. 또 다른 예로, 전달할 SKDC 발행 티켓과 같은 래핑된 데이터를 포함할 수 있다.

13.2.1.6 Private Data

Private Data 는 Device 를 인증하고, 데이터의 보호를 설정 또는 해제하거나 인증 시도 응답을 검증하는데 사용되는 비밀 정보를 포함한다.

Private Data 는 SRM 의 신뢰 컴퓨팅 기반 외부로 공개되지 않아야 한다. SRM 의 신뢰 컴퓨팅 기반의 구현에는 보안 요소 (SE) 또는 신뢰 실행 환경 (TEE)이 사용되어야 한다. 이러한 상황에서, Private Data 컨텐츠는 핸들이거나 보안 저장 resource 에의 참조이어야 한다.

13.2.1.7 Optional Data

Optional Data 는 선택적으로 제공되지만 키 관리, 확장성, 또는 성능 최적화의 편의를 도모하는 정보를 포함한다. 예를 들어, Credential Type 이 인증서를 식별하면, 이는 인증서 폐기 상태 값 및 상호 인증에 사용될 Certificate Authority (CA) 인증서를 포함한다.

13.2.1.8 Period

Period Property 는 크리덴셜의 유효 기간을 식별한다. 유효 기간이 지정되어 있지 않으면, 크리덴셜 수명은 정의되지 않는다. 날짜-시각 기능을 구현하지 않는 제한된 Device 는 CMS 로부터 현재 날짜-시각 정보를 취득한다.

13.2.1.9 크리덴셜 갱신 방법 타입 정의

oic.sec.crm 은 CMS 가 구현하는 크리덴셜 갱신 방법을 정의한다.

Value Type Name	Value Type URN	사용 가능한 크리덴셜 타입	설명
Provisioning Service	oic.sec.crm.pro	All	CMS 는 만료가 가까운 크리덴셜의 재 발행을 개시한다. Server 는 저장 resource 의 관리를 위해 만료된 크리덴셜을 삭제해야 한다. Resource Owner Property 는 프로비저닝 서비스를 참조한다. Server 는 /oic/sec/cred.rowneruuid Resource 를 사용해서 이 크리덴셜 갱신 방법을 지원하는 추가적인 키 관리 서비스를 식별한다.
Pre-shared Key	oic.sec.crm.psk	[1]	Server 는 크리덴셜 만료 전에 Diffie-Hellman 기반의 ciphersuite 와 현재 PSK 를 사용해서 DTLS 연결을 개시함으로써 ad-hoc 키 프레시를 수행한다. 신규 DTLS MasterSecret 값이 신규 PSK 로 된다. Server 는 신규 유효 기간을 선택한다. 신규 유효 기간 값은 현재 크리덴셜에 대해 유효 기간을 갱신하는 Device 로 전송된다. Device 는 성공 응답을 리턴함으로써 이 갱신을 받아들이거나 실패 응답을 리턴함으로써 이 갱신을 거부한다. Server 는 /oic/sec/cred.rowneruuid Resource 를 사용해서 이 크리덴셜 갱신 방법을 지원하는 키 관리 서비스를 식별한다.
Random PIN	oic.sec.crm.rdp	[16]	Server 는 oic.sec.crm.psk 접근에 이어서 ad-hoc 키 갱신을 수행하지만, 대역 외 채널을 통해 원격 Device 전달되는 random PIN 값을 생성하지 않는다. 현재 PSK + PIN 은 해시되어 DTLS ciphersuite 와 함께 사용될 신규 PSK'를 형성한다. 즉, PSK' = SHA256 (PSK, PIN). Server 는 /oic/sec/cred.rowneruuid Resource 를 사용해서 이 크리덴셜 갱신 방법을 지원하는 키 관리 서비스를 식별한다.
SKDC	oic.sec.crm.skdc	[1, 2, 4, 32]	Server 는 Device 에 대한 티켓을 취득하기 위한 요청을 발행한다. Server 는 티켓에 대한 응답에 포함된 정보를 사용해서 크리덴셜을 갱신한다. Server 는 /oic/sec/cred.rowneruuid Resource 를 사용해서 이 크리덴셜 갱신 방법을 지원하는 키 관리 서비스를 식별한다.
PKCS10	oic.sec.crm.pk10	[8]	Server 는 신규 인증서를 취득하기 위한 PKCS#10 인증서 요청 메시지를 발행한다. Server 는 /oic/sec/cred.rowneruuid Resource 를 사용해서 이 크리덴셜 갱신 방법을 지원하는 키 관리 서비스를 식별한다.

2829 **표 30 –oic.sec.crmtype Property 의 값 정의**

13.2.1.1 크리덴셜 용도

크리덴셜 용도는 Device 에 대해 크리덴셜을 사용해야 하는 상황을 나타낸다. 다섯 개의 값이 정의된다.

- oic.sec.cred.trustca: 이 인증서는 섹션 10.3 에 정의된 바와 같이 인증서 체인 검증을 목적으로 하는 신뢰 앵커이다.
- oic.sec.cred.cert: 이 credusage 는 섹션 10.3 에 정의된 바와 같이 Device 가 개인 키를 소유하고 보안 섹션에서 아이덴티티 인증에 개인 키를 사용하는 인증서에 대해 사용된다.

- oic.sec.cred.rolecert: 이 credusage 는 섹션 10.3.1 에 정의된 바와 같이 Device 가 개인 키를 소유하고 하나 이상의 role 을 주장하는데 개인 키를 사용하는 인증서에 대해 사용된다.

- oic.sec.cred.mfgtrustca: 이 인증서는 섹션 7.3.6 에 정의된 바와 같이 Manufacturer Certificate Based Owner Transfer Method 를 목적으로 하는 신뢰 앵커이다.
- oic.sec.cred.mfgcert: 이 인증서는 섹션 7.3.6 에 정의된 바와 같이 Device 가 개인 키를 소유하고 Manufacturer Certificate Based Owner Transfer Method 에서의 인증에 개인 키를 사용하는 인증서에 대해 사용된다.

13.2.2 키 형식

13.2.2.1 대칭 키 형식

대칭 키는 다음과 같은 형식을 갖는다.

명칭	값	형식	설명
Length	16	OCTET	Length 에 따르는 8 비트 octet 의 수를 지정한다.
Key	opaque	OCTET 배열	octet 의 16 바이트 배열. PSK 기능에의 입력으로 사용되면 Length 가 생략된다.

표 31 – 128 비트 대칭 키

명칭	값	형식	설명
Length	32	OCTET	Length 에 따르는 8 비트 octet 의 수를 지정한다.
Key	opaque	OCTET 배열	octet 의 32 바이트 배열. PSK 기능에의 입력으로 사용되면 Length 가 생략된다.

표 32 – 256 비트 대칭 키

13.2.2.2 비 대칭 키

주: 비 대칭 키 형식 설정은 본 버전의 스펙에서는 사용할 수 없다.

13.2.2.3 인증서와 비 대칭 키

키 형식 설정은 인증서 정의에 의해 정의된다.

13.2.2.4 비밀번호

기술 참조: 비밀번호 형식 설정은 본 버전의 스펙에서는 사용할 수 없다.

13.2.3 크리덴셜 갱신 방법 상세

13.2.3.1.1 프로비저닝 서비스

resource 소유자는 프로비저닝 서비스를 식별한다. Server 가 크리덴셜이 갱신을 필요로 하고 다른 방법이 적용되지 않거나 실패하는 것으로 판단하면, Server 는 만료 전에 크리덴셜의 재 프로비저닝을 요청한다. 크리덴셜이 만료되면 Server 는 Resource 를 삭제해야 한다.

13.2.3.1.2 사전 공유 키

이 모드를 사용해서, 현재 PSK 는 DTLS 에서 Diffie-Hellman 세션 키를 구성하는데 사용된다. TLS_PRF 는 신규 (갱신된) PSK 를 생성하는 키 유도 함수 (key derivation function: KDF)로 사용된다.

PSK = TLS_PRF(MasterSecret, Message, length);

- MasterSecret 는 위의 ciphersuite 중 하나를 사용한 DTLS 핸드세이크로부터 생기는 MasterSecret 값이다.
- Message 는 다음 값들의 연결이다.
    - RM – 갱신 방법 – 즉, "oic.sec.crm.psk"
    - Device ID_A 는 DTLS ClientHello 를 제공한 Device ID 의 스트링 표현이다.
    - Device ID_B 는 DTLS ClientHello 메시지에 응답하는 Device 이다.
- length 는 바이트 단위의 메시지 길이이다.

Server 와 Client 는 둘 다 PSK 를 사용해서 /oic/sec/cred Resource 의 privatedata Property 를 갱신한다. Server 가 크리덴셜 갱신을 개시하면, 신규 유효 기간을 선택한다. Server 는 선택한 유효 기간을 새롭게 구성된 DTLS 세션을 통해 Client 에게 전송하여, Server 에 대해 대응하는 크리덴셜 Resource 를 갱신할 수 있다.

13.2.3.1.3 Random PIN

이 모드를 사용해서, 현재 만료되지 않은 PIN 은 RFC2898 에 따르는 PSK 를 생성하는데 사용된다. PSK 는 신규 세션 키를 생성하기 위한 Diffie-Hellman 교환 동안에 사용된다. 세션 키는 PIN 에서 PSK 모드로 전환하는데 사용되어야 한다.

PIN 은 Server 에 의해 무작위로 생성되어 대역 외 방법을 통해 Client 로 전달된다. 사용되는 OOB 방법은 본 스펙의 범위에 포함되지 않는다.

의사 난수 함수 (pseudo-random function: PBKDF2)는 RFC2898 에 의해 정의된다. PIN 은 사전 공유 키를 생성하는데 사용되는 공유 값이다. PIN 인증 사전 공유 키 (PPSK)는 PSK 를 허용하는 DTLS ciphersuite 로 제공된다.

PPSK = PBKDF2(PRF, PIN, RM, Device ID, c, dkLen)

PBKDF2 함수는 다음과 같은 파라미터를 갖는다.

- PRF - DTLS PRF 를 사용한다.
- PIN - Devices 간에 공유된다.
- RM – 갱신 방법 – 즉, "oic.sec.crm.rdp"
- Device ID - 신규 Device 의 UUID.
- c - 1000 으로 초기화된 반복 카운트, 사용할 때마다 증가.
- dkLen - 도출된 PSK 의 octet 단위의 원하는 길이.

Server 와 Client 는 둘 다 PPSK 를 사용해서 /oic/sec/cred Resource 의 PrivateData Property 를 갱신한다. Server 가 크리덴셜 갱신을 개시하면, 신규 유효 기간을 선택한다. Server 는 선택한 유효 기간을 새롭게 구성된 DTLS 세션을 통해 Client 에게 전송하여, Server 에 대해 대응하는 크리덴셜 Resource 를 갱신할 수 있다.

13.2.3.1.4 SKDC

DTLS 세션이 oic.sec.crm.skdc 크리덴셜 갱신 방법을 지원하는 svctype="oic.sec.cms"를 가진 /oic/sec/cred.rowneruuid 에 대해 개방된다. oic.sec.cms 서비스에 대해 티켓 요청 메시지가 전달되고, 응답으로 티켓 요청이 리턴된다. Server 는 티켓 응답 컨텐츠에 의해 안내되는 /oic/sec/cred Resource 를 갱신하거나 인스턴스화한다.

13.2.3.1.5 PKCS10

DTLS 세션이 oic.sec.crm.pk10 크리덴셜 갱신 방법을 지원하는 svctype="oic.sec.cms"를 가진 /oic/sec/cred.rowneruuid 에 대해 개방된다. 서비스에 대해 PKCS10 형식의 메시지가 전달된다. 갱신된 인증서의 발행 후에, oic.sec.cms 서비스는 Server 에 대해 인증서를 전달한다. Server 는 인증서 컨텐츠에 의해 안내되는 /oic/sec/cred Resource 를 갱신하거나 인스턴스화한다.

13.2.3.2 Resource 소유자

Resource Owner Property 는 Device 온보딩 직후에 지원 서비스에의 액세스가 확립되기 전에 크리덴셜 프로비저닝이 발생하도록 한다. 이는 Device 복원 상황에 대한 응답으로 /oic/sec/cred Resource 를 관리하도록 인가된 개체를 식별한다.

13.3 인증서 폐기 목록

13.3.1 CRL Resource 정의

Device 인증서와 개인 키는 cred Resource 내에 보관된다. CRL 은 폐기 목록을 유지하기 위해 새롭게 정의된 별도의 crl Resource 를 사용해서 유지 및 갱신된다.

고정 URI	Resource Type Title	Resource Type ID ("rt" 값)	Interface	설명	관련 기능 연동
/oic/sec/crl	CRLs	urn:oic.r.crl	baseline	Device 인증서 폐기에 대한 CRL 을 포함하는 Resource	보안

2920 **표 33 –oic.r.crl Resource 정의**

Property Title	Property Name	Value Type	Value Rule	액세스 모드	필수	설명
CRL Id	crlid	UINT16	0 – 64K-1	RW	예	다른 Resource 로부터의 참조를 위한 CRL ID
This Update	thisupdate	스트링	-	RW	예	이것은 CRL 이 갱신된 시간을 나타낸다. (UTC)
CRL Data	crldata	스트링	-	RW	예	CRL 프로파일 내의 CertificateList 를 토대로 한 CRL 데이터.

2921 **표 34 –oic.r.crl Resource 의 Property**

2922 ### 13.4 ACL Resource

2923 Server 에 의해 호스트되는 모든 Resource 는 ACL 정책에 매칭되어야 한다. ACL 정책은
2924 /oic/sec/acl, /oic/sec/amacl, 및 /oic/sec/sacl 의 세 가지 ACL Resource Type 을 사용해서 표현할
2925 수 있다. Resource 에의 액세스를 요청하는 주체 (예: Client 의 Device ID)는 ACL 체크의 적용 전에
2926 인증된다. 모두가 사용할 수 있는 Resource 는 와일드카드 주체 참조를 사용할 수 있다. 비 보안
2927 통신 채널을 통해 액세스 가능한 모든 Resource 는 와일드카드 주체를 사용해서 명명된다.

2928

2929 #### 13.4.1 OCF 액세스 제어 목록 (ACL) BNF 정의 ACL 구조

2930 Backus-Naur Form (BNF) 표기법에서의 ACL 구조:

<ACL>	<ACE> {<ACE>}
<ACE>	<SubjectId> <ResourceRef> <Permission> {<Validity>}
<SubjectId>	<DeviceId> | <Wildcard> | <RoleId>
<DeviceId>	<UUID>
<RoleId>	[<Authority>] <RoleName> {<RoleName>}

<RoleName>	<URI>
<Authority>	<UUID>
<ResourceRef>	' (' <OIC_LINK> {',' {OIC_LINK>} ')'
<Permission>	('C' | '-') ('R' | '-') ('U' | '-') ('D' | '-') ('N' | '-')
<Validity>	<Period> {<Recurrence>}
<Wildcard>	'*'
<URI>	RFC3986 // **다음의 문헌은,** 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다. OIC Core Specification 정의
<UUID>	RFC4122 // **다음의 문헌은,** 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다. OIC Core Specification 정의
<Period>	RFC5545 기간
<Recurrence>	RFC5545 반복
<OIC_LINK>	JSON Schema 내의 **다음의 문헌은,** 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다. OIC Core Specification 정의

표 35 –OCF ACL 의 BNF 정의

<DeviceId> 토큰은 요청자가 요청자를 <ACE> 정책에 매칭시키기 위해 자신의 아이덴티티로 <UUID>를 사용하는 크리덴셜을 소유해야 하는 것을 의미한다.

<RoleID> 토큰은 요청자가 요청자를 <ACE> 정책에 매칭시키기 위해 자신의 role 로서 <URI>를 갖는 role 크리덴셜을 소유해야 하는 것을 의미한다.

<Wildcard> 토큰 "*"는 인증의 유무에 관계 없이 모든 요청자가 <ACE> 정책에 매칭되는 것을 의미한다.

<SubjectId>가 <ACE> 정책에 매칭되면, <ACE> 정책을 resource 에 매칭시키는데 <ResourceRef>가 사용된다.

<OIC_LINK> 토큰은 호스트된 resource 의 존재를 질의하는데 사용되는 값을 포함한다.

<Permission> 토큰은 <SubjectId>와 <ResourceRef> 매칭이 빈 매칭 집합을 생성하지 않는 한 <ACE> 정책에 의해 부여된 권한을 지정한다.

권한은 CREATE ('C'), RETRIEVE ('R'), UPDATE ('U'), DELETE ('D'), NOTIFY ('N'), 및 NIL ('-')에 관해서 정의된다. NIL 은 해당하는 권한이 부여되지 않음을 의미하는 권한 문자로 대체된다.

빈 매칭 집합 결과는 기본적으로 아무런 액세스 권한도 부여되지 않음을 의미한다.

<Validity> 토큰이 존재하면, 부여된 <Permission>이 시간 <Period>에 제한된다. <Validity>는 패턴에 따라 액세스가 교대로 부여되고 취소되는 <Recurrence> 패턴으로 나뉘어질 수 있다.

13.4.2 ACL Resource

ACL 에는 'acl'과 'acl2'의 두 가지 타입이 있다. 'acl'은 타입 'ace'의 목록이고, 'acl2'는 타입 'ace2'의 목록이다. Device 는 /acl Resource 를 호스트하지 않는다. 주: /acl Resource 는 역 호환성 및 Provisioning Tool 등에 의한 사용에 대해 정의된다.

"aces2"의 관리 (/oic/sec/acl2 Resource 에 대한)를 허용하는 interface 를 제공하기 위해, oic.r.ace2 Resource 에 대한 Array Property, RETRIEVE, UPDATE, 및 DELETE 동작은 다음과 같이 수행된다.

1. RETRIEVE 는 전체 Resource 표현을 리턴한다.
2. UPDATE 는 다음과 같이 UPDATE 요청과 함께 전송된 표현에 포함된 Property 에 대해 대체 또는 추가한다.
    a. UPDATE 표현이 배열 Property 를 포함하면,
        i. Supplied s with an that matches an 기존의 "aceid"와 일치하는 "aceid"로 제공된 ACE 는 기존의 "aces2" 배역의 대응하는 ACE 를 완전히 대체한다.
        ii. "aceid" 없이 제공된 ACE 는 기존의 "aces2" 배열에 추가되고, Server 에 의해 고유한 (acl2 Resource 에 대해) "aceid"가 생성되어 신규 ACE 에 할당된다.

2965 interface 의 결정성을 향상시키고 경합 조건을 완화시키기 위해 삭제된 ACE 의
2966 "aceid"는 재 사용하지 않는다.

2967

2968 iii. 기존의 "aceid"와 일치하지 않는 "aceid"로 제공된 ACE 는 제공된 "aceid"를
2969 사용해서 기존의 "aces2" 배열에 추가된다.

2970 3. 질의 파라미터가 없는 DELETE 는 전체 "aces2" 배열을 삭제하지만, oic.r.ace2 Resource 는
2971 삭제하지 않는다.

2972 4. 하나 이상의 "aceid" 질의 파라미터를 갖는 DELETE 는 "aces2" 배열로부터 대응하는 aceid 를
2973 갖는 ACE 를 삭제한다.

2974 모든 ACL Resource 가 질의되고 요청자로부터 요청된 Resource 에 대해 더 이상 개체를 찾을 수
2975 없으면 – 예를 들어, /oic/sec/acl, /oic/sec/sacl, 및 /oic/sec/amacl 가 주체 및 요청된 Resource 와
2976 매칭되지 않으면, 로컬 ACL Resource 의 평가가 완료된다.

2977 액세스 매니저 ACL 이 요청을 만족하면, Server 는 AMS 에의 보안 연결을 개방한다. 1 차 AMS 를
2978 사용할 수 없을 때는 2 차 AMS 를 시도해야 한다. Server 는 주체 및 요청된 resource 를 필터
2979 조건으로 제공하여 AMS 에 대해 질의한다. Server Device ID 는 AMS 에 의해 보안 연결
2980 컨텍스트로부터 취득되어 필터 조건에 포함된다. AMS 정책이 만족되면 Permission Property 가
2981 리턴된다.

2982 요청된 Resource 가 매칭되지 않으면, Server 는 에러를 리턴한다. 요청자는 구성된 AMS 서비스를
2983 발견하기 위해 Server 에 질의해야 한다. Client 는 AMS 에 대해 sacl (/oic/sec/sacl) Resource 를
2984 요청해야 한다. 다음과 같은 동작을 수행함으로써 이러한 유형의 요청을 구현할 수 있다.

2985

2986 1. Client: AMS 에 대해 보안 연결을 개방한다.

2987 2. Client: /oic/sec/acl?device="urn:uuid:XXX...",resource="URI"를 취득한다.

2988 3. AMS: AMS 에 의해 서명된 /oic/sec/sacl Resource 를 구성하고, GET 명령어에 대한
2989 응답으로 이를 리턴한다.

2990 4. Client: /oic/sec/sacl [{ ...sacl... }]를 투고한다.

2991 5. Server: AMS 크리덴셜을 사용해서 sacl 서명을 검증하고, 유효하면 ACL Resource 를
2992 설치한다.

2993 6. Client: 원래의 Resource 액세스 요청을 재 시도한다. 이 때 로컬 acl 평가에는 신규 ACL 이
2994 포함된다.

/oic/sec/sacl Resource 에 포함된 ACL 은 반복된 Resource 요청을 만족하도록 더 긴 시간의 액세스를 부여해야 한다.

고정 URI	Resource Type Title	Resource Type ID ("rt" 값)	Interface	설명	관련 기능 연동
/oic/sec/acl	ACL	urn:oic.r.acl	baseline	액세스 관리용 Resource	보안

2997 **표 36 –oic.r.acl Resource 정의**

Property Title	Property Name	Value Type	Value Rule	필수	액세스 모드	Device 상태	설명
ACE List	aclist	oic.sec.ace	-	예		-	ACL resource 내의 Access Control Entry 목록. 이 Property 는 "aces", oic.sec.ace1 resource 의 배열과 "aces2", 및 oic.sec.ace2 Resource 의 배열을 포함한다.
					R	RESET	Server 는 제조사 기본설정으로 설정한다.
					RW	RFOTM	보안 세션이 구성된 후에 OBT 가 oic.sec.ace2 항목을 구성 및 선택한다.
					RW	RFPRO	상호 인증된 보안 세션이 구성되면, AMS (rowneruuid property 를 통해 참조)가 oic.sec.ace2 항목을 갱신한다. vertical resource 에의 액세스는 금지된다.
					R	RFNOP	매칭되는 ACE 를 찾으면 vertical resource 에의 액세스가 허용된다.
					RW	SRESET	보안 세션이 구성되고 Server 와 OBT 가 인증되면, OBT (devowneruuid property 를 통해 참조)는 oic.sec.ace2 항목의 무결성을 평가하고 이를 갱신해야 한다.

Resource Owner ID	rowneruuid	스트링	uuid	예	-	-	Server 가 신뢰하는 서비스 제공자를 참조하기 위해 resource owner property (rowneruuid)를 사용한다. Server 는 서비스 제공자가 요청된 동작을 수행하도록 인가되었는지를 검증한다.
					R	RESET	Server 는 nil uuid 값 (예: "00000000-0000-0000-0000-000000000000")으로 설정한다.
					RW	RFOTM	소유권 이전 세션이 성공적으로 구성되면 DOXS 는 /acl.rowneruuid 및 /acl2.rowneruuid property 를 구성해야 한다.
					R	RFPRO	해당 없음
					R	RFNOP	해당 없음
					RW	SRESET	상호 인증된 보안 세션이 구성되면 DOXS (/doxm.devowneruuid property 를 통해 참조)는 resource owner property 를 검증하고, 필요하면 갱신해야 한다. Rowneruuid 가 유효한 DOXS 를 참조하지 않으면 Server 는 RESET Device 상태로 전환한다..

2998 **표 37 –oic.r.acl Resource 의 Property**

Property Title	Property Name	Value Type	Value Rule	액세스 모드	필수	설명
Resources	resources	oic.oic-link	배열		예	보안 정책이 적용되는 애플리케이션의 Resource.
Permission	permission	oic.sec.crudntype	bitmask	RW	예	CRUDN 권한의 비트마스크 인코딩

Validity	validity	oic.sec.ace/definitions/time-interval	배열	RW	아니오	기간과 반복의 튜플의 배열. 이 배열의 각 항목은 RFC5545 Period 를 사용한 기간을 나타내는 스트링 및 RFC5545 Recurrence 를 사용한 반복 규칙을 나타내는 스트링 배열을 포함한다.
Subject ID	subjectuuid	스트링	uuid, "*"	RW	예	이 ACE 가 적용되는 Device 또는 익명 액세스용 "*"를 식별하기 위한 uuid.

2999 **표 38 –oic.r.ace Property 의 속성**

값	액세스 정책	설명	주
bx0000,0000 (0)	No permissions	권한 없음	
bx0000,0001 (1)	C	CREATE	
bx0000,0010 (2)	R	RETREIVE, OBSERVE, DISCOVER	"R" 허가 비트는 Read 허가와 Observe 허가를 둘 다 관장한다.
bx0000,0100 (4)	U	WRITE, UPDATE	
bx0000,1000 (8)	D	DELETE	
bx0001,0000 (16)	N	NOTIFY	OCF 1.0 에서는 "R"이 Observe 허가를 관장하므로 "N" 허가 비트가 무시된다. 이는 본 스펙의 향후 버전에서 문서화한다.

3000 **표 39 –oic.sec.crudntype Property 의 값 정의**

고정 URI	Resource Type Title	Resource Type ID ("rt" 값)	Interface	설명	관련 기능 연동
/oic/sec/acl2	ACL2	oic.r.acl2	baseline	액세스 관리용 Resource	보안

3001 **표 40 –oic.sec.acl2 Resource 정의**

Property Name	Value Type	필수	Device 상태	액세스 모드	설명
aclist2	oic.sec.ace2 의 배열				aclist2 property 는 "oic.sec.ace2" 타입의 ACE 레코드의 배열이다. Server 는 로컬 resource 에 액세스 제어를 적용하는데 이 목록을 사용한다.
			RESET	R	Server 는 제조사 기본설정으로 설정한다.
			RFOTM	RW	보안 세션이 구성된 후에 OBT 가 oic.sec.ace2 항목을 구성 및 선택한다.
			RFPRO	RW	상호 인증된 보안 세션이 구성되면, AMS (rowneruuid property 를 통해 참조)가 oic.sec.ace2 항목을 갱신한다. vertical resource 에의 액세스는 금지된다.
			RFNOP	R	매칭되는 ACE 를 찾으면 vertical resource 에의 액세스가 허용된다.
			SRESET	RW	보안 세션이 구성되고 Server 와 OBT 가 인증되면, OBT (devowneruuid property 를 통해 참조)는 oic.sec.ace2 항목의 무결성을 평가하고 이를 갱신해야 한다.
rowneruuid	uuid	Yes			oic.sec.acl 의 rowneruuid 와 동일.

3002 **표 41 –oic.sec.acl2 Resource 의 Property**

3003

Property Name	Value Type	필수	설명
subject	oic.sec.roletype, oic.sec.didtype, oic.sec.conntype	예	role, Device ID, 또는 연결 타입이 일치할 때 Client 는 ACE 의 주체이다.
resources	oic.sec.ace2.resource-ref 의 배열	예	보안 정책이 적용되는 애플리케이션의 resource.
permission	oic.sec.crudntype.bitmask	예	CRUDN 권한의 비트마스크 인코딩
validity	oic.sec.time-pattern 의 배열	아니오	기간과 반복의 튜플의 배열. 이 배열의 각 항목은 RFC5545 Period 를 사용한 기간을 나타내는 스트링 및 RFC5545 Recurrence 를 사용한 반복 규칙을 나타내는 스트링 배열을 포함한다.
aceid	정수	예	aceid 는 'aces' 배열에 대해 고유하다.

3004 **표 42 – oic.sec.ace2 data type 정의**

Property Name	Value Type	필수	설명
href	uri	아니오	ACE 가 적용되는 resource 를 참조하는 URI.
rt	스트링의 배열	아니오	ACE 가 적용되는 resource type.
if	스트링의 배열	아니오	ACE 가 적용되는 interface.
wc	스트링	아니오	와일드카드 매칭 정책: "+" – 모든 발견 가능 resource 에 매칭된다. "-" – 모든 발견 불가능 resource 에 매칭된다. "*" – 모든 resource 에 매칭된다.

3005 **표 43 – oic.sec.ace2.resource-ref data type 정의**

Property Name	Value Type	Value Rule	설명
conntype	스트링	enum ["auth-crypt", "anon-clear"]	이 property 는 ACE 가 연결 또는 메시지 보호 타입을 토대로 매칭되도록 한다.
		auth-crypt	Client 가 인증되고 데이터 채널 또는 메시지가 암호화되고 무결성이 보호되는 경우 ACE 가 적용된다.
		anon-clear	Client 가 인증되지 않고 데이터 채널 또는 메시지가 암호화되지 않았지만 무결성이 보호되는 경우 ACE 가 적용된다.

3006 **표 44 – oic.sec.conntype Property 의 값 정의**

3007 로컬 ACL Resource 는 OCF 스택 인스턴스 내에서 Resource 액세스 시행점에 대해 정책을
3008 제공한다. OCF framework 는 Client 에게 Server Resource 에 액세스할 수 있는 문을 제공한다.
3009 이는 ACL 의 정책을 사용해서 주체의 요청을 평가한다.

3010 ACL 정책 내에서 명명된 Resource 는 전체 또는 부분적이어야 한다. 전체 Resource reference 는
3011 Resource 를 호스트하는 원격 Device 의 Device ID 를 포함해야 한다. 부분 resource reference 는
3012 로컬 Resource Server 가 Resource 를 호스트함을 의미한다. 전체 resource reference 가 주어지면,
3013 액세스를 시행하는 Intermediary 가 Resource Server 에의 보안 채널을 갖고, Resource Server 는
3014 Intermediary 가 Resource 액세스 시행점으로 대신 동작하도록 인가되었는지를 검증한다.

3015

3016 Resource Server 는 액세스 시행이 적용되는 Device 와 ACL Resource 에의 참조를 포함해야 한다.
3017 그러나, Server Resource 에의 액세스가 이미 부여된 상태일 것이므로 액세스 제어 처리를 위해
3018 액세스 시행 로직이 이들 참조에 의존하지는 않는다.

3019

3020 로컬 ACL Resource 는 이러한 Resource 를 인스턴스화 및 수정하도록 인가된 Resource Owner
3021 서비스를 식별한다. 이는 다른 ACL Resource 에의 비 종단 의존성을 방지한다. 그럼에도 불구하고,
3022 ACL Resource 를 사용해서 ACL Resource 에의 액세스 권하는 부여하는 것이 바람직하다.

ACE 또는 ACE2 항목이 요청 시각을 포함하면 ACE 또는 ACE2 항목을 *현재 유효한* 것으로 본다. ACE 또는 ACE2 내의 유효 기간은 반복되는 시간인 경우가 있다 (예: 매일 1:00-2:00). 요청에 지정된 resource 의 ACE 또는 ACE2 resource property 에의 매칭은 섹션 **Error! Reference source not found.**에 정의된다. 예를 들어, 하나의 매칭 방식은 요청 내의 Resource URI 가 ACE 또는 ACE2 항목 내의 resource reference 중 하나와 정확하게 일치하는 것이다.

다음 중 어느 하나가 참이면 요청이 ACE 와 매칭된다.

1. 보안 세션에 관련된 deviceuuid 가 ACE 의 "subjectuuid"와 일치하고, 요청 resource 가 ACE 의 "resource" 중 하나와 일치하고, ACE 가 현재 유효한 상태이다.

2. ACE "subjectuuid"가 와일드카드 "*" 문자를 포함하고, 요청 resource 가 ACE 의 "resource" 중 하나와 일치하고, ACE 가 현재 유효한 상태이다.

3. 인증이 대칭 키 크리덴셜을 사용할 때,

   요청의 CoAP 페이로드 질의 스트링이 role 을 지정하고, 이는 현재 보안 세션의 대칭 키 크리덴셜과 연관되고,

   요청의 CoAP 페이로드 질의 스트링이 role 을 지정하고, 이는 현재 보안 세션의 oic.r.cred.creds.roleid property 에 포함되고,

   요청 resource 가 ACE 의 "resource" 중 하나와 일치하고,

   ACE 가 현재 유효한 상태이다.

다음 중 어느 하나가 참이면 요청이 ACE2 와 매칭된다.

1. ACE2 "subject"가 oic.sec.didtype 타입이고, 보안 세션과 관련된 deviceuuid 와 일치하는 UUID 값을 갖고,

   요청 resource 가 ACE2 oic.sec.ace2.resource-ref 의 "resource" 중 하나와 일치하고,

   ACE2 가 현재 유효한 상태이다.

2. ACE2 "subject"가 oic.sec.conntype 타입이고, 현재 구성된 연결 타입과 일치하는 와일드카드 값을 갖고,

   요청 resource 가 ACE2 oic.sec.ace2.resource-ref 의 "resource" 중 하나와 일치하고,

   ACE2 가 현재 유효한 상태이다.

3. Client 인증이 인증서 크리덴셜을 사용할 때,

   role 인증서에 포함된 roleid 값 중 하나가 ACE2 oic.sec.roletype 의 "roleid"와 일치하고,

   role 인증서 공개 키가 현재 보안 세션의 구성에 사용된 인증서의 공개 키와 일치하고,

   요청 resource 가 ACE2 oic.sec.ace2.resource-ref 의 "resource" 중 하나와 일치하고,

   ACE2 가 현재 유효한 상태이다.

4. Client 인증이 인증서 크리덴셜을 사용할 때,

   요청의 CoAP 페이로드 질의 스트링이 role 을 지정하고, 이는 role 인증서에 포함된 role 집합의 구성요소이고,

   role 인증서에 포함된 roleid 값이 ACE2 oic.sec.roletype 의 "roleid"와 일치하고,

   role 인증서 공개 키가 현재 보안 세션의 구성에 사용된 인증서의 공개 키와 일치하고,

   요청 resource 가 ACE2 oic.sec.ace2.resource-ref 의"resource" 중 하나와 일치하고,

   ACE2 가 현재 유효한 상태이다.

5. Client 인증이 대칭 키 크리덴셜을 사용할 때,

보안 세션에 사용된 대칭 키 크리덴셜과 관련된 roleid 값 중 하나가 ACE2 oic.sec.roletype 의 "roleid"와 일치하고,

요청 resource 가 ACE2 oic.sec.ace2.resource-ref 의"resource" 중 하나와 일치하고,

ACE2 가 현재 유효한 상태이다..

6. Client 인증이 대칭 키 크리덴셜을 사용할 때,

요청의 CoAP 페이로드 질의 스트링이 role 을 지정하고, 이는 현재 보안 세션의 oic.r.cred.creds.roleid property 에 포함되고,

요청의 CoAP 페이로드 질의 스트링이 ACE2 oic.sec.roletype 의 "roleid"와 일치하는 role 을 지정하고,

요청 resource 가 ACE2 oic.sec.ace2.resource-ref 의"resource" 중 하나와 일치하고,

ACE2 가 현재 유효한 상태이다.

'매칭되는' ACE 중 어느 하나가 요청을 허용하는 권한을 포함하면 요청이 승인된다. 그렇지 않으면, 요청이 거절된다.

ACE 가 resource 에의 권한을 명시적으로 거절할 수 있는 방법은 없다. 그러므로 주어진 role 을 가진 Device 가 동일한 role 을 가진 또 하나의 Device 와 약간 다른 권한을 갖는 경우, 이들은 서로 다른 role 로 프로비저닝되어야 한다.

13.5 Access Manager ACL Resource

고정 URI	Resource Type Title	Resource Type ID ("rt" 값)	Interface	설명	관련 기능 연동
/oic/sec/amacl	Managed ACL	urn:oic.r.amacl	baseline	액세스 관리용 Resource	보안

표 45 –oic.r.amacl Resource 정의

Property Title	Property Name	Value Type	Value Rule	액세스 모드	필수	설명
Resources	resources	oic.sec.ace2	배열	RW	예	이 호스트의 Resource 에 대한 복수의 링크

표 46 –oic.r.amacl Resource 의 Property

13.6 서명된 ACL Resource

고정 URI	Resource Type Title	Resource Type ID ("rt" 값)	Interface	설명	관련 기능 연동
/oic/sec/sacl	Signed ACL	urn:oic.r.sacl	baseline	액세스 관리용 Resource	보안

표 47 –oic.r.sacl Resource 정의

Property Title	Property Name	Value Type	Value Rule	액세스 모드	필수	설명
ACE List	aclist2	oic.sec.ace2	배열	RW	예	ACL Resource 내의 액세스 제어 항목
Signature	signature	oic.sec.sigtype	-	RW	예	ACL Resource 에 대한 서명

표 48 –oic.r.sacl Resource 의 Property

Property Title	Property Name	Value Type	Value Rule	단위	액세스 모드	필수	설명

Signature Type	sigtype	스트링	-	-	RW	예	사전 정의된 서명 형식을 지정하는 스트링. "oic.sec.sigtype.jws" – RFC7515 JSON 웹 서명 (JWS) 객체 "oic.sec.sigtype.pk7" – RFC2315 base64 인코딩된 객체 "oic.sec.sigtype.cws" – CBOR 인코딩된 JWS 객체
Signature Value	sigvalue	스트링	-	-	RW	예	인코딩된 서명

표 49 –oic.sec.sigtype Property 의 속성

13.7 프로비저닝 상태 Resource

/oic/sec/pstat Resource 는 Device 프로비저닝 상태를 보유한다. Device 프로비저닝은 Client 지향 또는 Server 지향이어야 한다. Client 지향 프로비저닝은 Client Device 에 의존해서 Server Resource 의 인스턴스화 및 갱신에 관한 대상, 때, 및 방법을 결정한다. Server 지향 프로비저닝은 Server 에 의존해서 조건에 따라 프로비저닝을 시도한다. Server 지향 프로비저닝은 /oic/sec/cred.rowneruuid 및 /oic/sec/cred Resource 의 구성에 의존해서 최소한 적절한 지원 서비스와 더불어 보안 연결을 개방하는데 필요한 설정으로 Server 를 부트스트랩한다.

고정 URI	Resource Type Title	Resource Type ID ("rt" 값)	Interface	설명	관련 기능 연동
/oic/sec/pstat	3) Provisioning Status	urn:oic.r.pstat	baseline	Device 프로비저닝 상태 관리용 Resource	구성

3106 **표 50 –oic.r.pstat Resource 정의**

Property Title	Property Name	Value Type	Value Rule	필수	액세스 모드	Device 상태	설명
Device Onboarding State	dos	oic.sec.dostype	-	예	RW		Device 온보딩 상태
Is Operational	isop	Boolean	T|F	예	R	RESET	Cm 이 영이 아니더라도 Device 가 기능할 수 있다. Device IsOp 가 FALSE 일 때 Tm 을 만족시키는데 관련된 요청만 제공한다. Server 는 FALSE 로 설정된다.
					R	RFOTM	Server 는 FALSE 로 설정된다.
					R	RFPRO	Server 는 FALSE 로 설정한다.
					R	) RFNOP	Server 는 TRUE 로 설정한다.
					R	) SRESET	Server 는 FALSE 로 설정한다.
Current Mode	cm	oic.sec.dpmtype	bitmask	예	R		Server 는 0000,0001 로 설정한다.
					R		OTM 이 성공적으로 완료되면 DOXS 에 의해 00xx,xx10 으로 설정된다.

					R		인증이 성공적으로 완료된 후에 CMS, AMS, DOXS 에 의해 설정된다.
					R		인증이 성공적으로 완료된 후에 CMS, AMS, DOXS 에 의해 설정된다.
					R		Server 는 0000,0001 로 설정한다.
Target Mode	tm	oic.sec.dpmtype	bitmask	아니오	R		Server 는 0000,0010 으로 설정한다.
					RW		OTM 이 성공적으로 완료되면 DOXS 에 의해 설정된다.
					RW		인증이 성공적으로 완료되면 CMS, AMS, DOXS 에 의해 설정된다.
					RW		인증이 성공적으로 완료되면 CMS, AMS, DOXS 에 의해 설정된다.
					RW		오류로부터 복구하는데 필요 시에 DOXS 에 의해 설정된다. Server 는 SRESET 으로 전환 시에 XXXX,XX00 으로 설정한다.
Operational Mode	om	oic.sec.pomtype	bitmask	예	R		Server 는 제조사 기본설정으로 설정한다.
					RW		OTM 이 성공적으로 완료되면 DOXS 에 의해 설정된다.

					RW		인증이 성공적으로 완료되면 CMS, AMS, DOXS 에 의해 설정된다.
					RW		인증이 성공적으로 완료되면 CMS, AMS, DOXS 에 의해 설정된다.
					RW		DOXS 에 의해 설정된다.
Supported Mode	sm	oic.sec.pomtype	bitmask	예	R		지원되는 프로비저닝 서비스 운영 모드
Device UUID	deviceuuid	스트링	uuid	예	RW		[DEPRECATED] 상태가 적용되는 Device 를 식별하기 위한 uuid.
Resource Owner ID	rowneruuid	스트링	uuid	예	R	RESET	Server 는 nil uuid 값 (예: "00000000-0000-0000-0000-000000000000")으로 설정한다.
					RW	RFOTM	소유권 이전 세션이 성공적으로 구성되면 DOXS 는 /pstat.rowneruuid property 를 구성해야 한다.
					R	RFPRO	해당 없음
					R	RFNOP	해당 없음

					RW	SRESET	DOXS (/doxm.devowneruuid property 를 통해 참조)는 상호 인증된 보안 세션이 구성되면 resource owner property 를 검증하고, 필요하면, 갱신해야 한다. Rowneruuid 가 유효한 DOXS 를 참조하지 않으면 Server 는 RESET Device 상태로 전환한다.

표 51 –oic.r.pstat Resource 의 Property

프로비저닝 상태 Resource /oic/sec/pstat 는 Device 가 자기 지향 프로비저닝을 수행할 수 있도록 하는데 사용된다. Device 는 자신의 현재 구성 상태와 타겟 구성 목표를 인지한다. 현재 상태와 목표 상태에 차이가 있으면, Device 는 /oic/sec/cred Resource 의 rowneruuid Property 를 참조해서 적절한 프로비저닝 서비스가 존재하는지 파악해야 한다. Device 는 그렇게 하도록 구성되어 있으면 프로비저닝을 요청해야 한다. /oic/sec/pstat Resource 의 om Property 는 이러한 상황에서 예상되는 Device 의 동작을 규정한다.

자기 지향 프로비저닝은 네트워크에서 단일 장애점이 되는 중앙 프로비저닝 기관에의 의존성을 최소화하기 위해 Device 가 더 큰 자율성을 갖고 기능하도록 한다.

Property Title	Property Name	Value Type	Value Rule	필수	액세스 모드	Device 상태	설명
Device Onboarding State	s	UINT16	enum (0=RESET, 1=RFOTM, 2=RFPRO, 3=RFNOP, 4=SRESET	예	R	RESET	Device가 Hard Reset 상태로 되어 있다.
					RW	RFOTM	Device가 Ready-For-Owner-Transfer-Method 상태로 되어 있다. OTM이 성공적으로 완료되면 DOXS에 의해 RFPRO로 설정된다.
					RW	RFPRO	Device가 Ready-For-PROvisioning 상태로 되어 있다. 인증이 성공적으로 완료되면 CMS, AMS, DOXS에 의해 설정된다.
					RW	RFNOP	Device Ready-For-Normal-Operation 상태로 되어 있다. 인증이 성공적으로 완료되면 CMS, AMS, DOXS에 의해 설정된다.
					RW	SRESET	Device가 Soft-Reset 상태로 되어 있다. 호출자가 인증되고 dos.s property를 갱신하도록 인가되면, /pstat.dos.s에 기록함으로써 원격으로 상태 전환을 수행할 수 있다. 인증이 성공적으로 완료되면 CMS, AMS, DOXS에 의해 설정된다.

Pending state	p	Boolean	T | F	예	R		TRUE (1) –Device resource 에 필요한 모든 변경이 완료될 때까지 's' 상태가 보류된다. FALSE (0) – 's' 상태 변경이 완료되었다.

표 52 –oic.sec.dostype Property 의 속성

모든 상태에서,

- /pstat.dos.p Property 는 모든 요청자에 대해 읽기 전용이다.
- 인증된 client 는 pstat.dos.s=<device_state>를 갱신함으로써 Device 상태 변경을 수행할 수 있다. 그렇게 함으로써 Server 가 지정된 Device 상태로 전환하는데 필요한 모든 변경을 자동으로 수행하도록 지시한다. 복수의 단계가 있을 수 있으므로 첫 번째 단계로 dos.p 값이 TRUE 로 설정되고 모든 단계가 완료될 때까지 TRUE 를 유지한다. 마지막 단계는 dos.p 를 FALSE 로 설정한다. dos.p 값이 FALSE 로 설정됨과 동시에 dos.s 값이 지정된 Device 상태로 설정된다. client 는 Device 상태 변경이 완료되면 /pstat.dos 가 통지되는 것을 관측할 수 있다.
- 적절한 role (예: DOXS, CMS, AMS)를 소유하고 있으면, Client 는 /pstat.dos.s 에 기록하도록 인증된다.
- 요청자가 도달할 수 없는 상태로의 전환을 시도하는 /pstat.dos.s 에의 쓰기 요청은 DEVICE_STATE_NOT_PERMITTED 에러를 초래한다.
- /pstat.dos.p 가 TRUE 일 때 /pstat.dos.s 에의 쓰기 요청은 DEVICE_STATE_NOT_READY 에러를 초래한다.
- 들어갈 때 /pstat.dos.p 는 TRUE 로 설정되어야 한다.
- 나올 때 /pstat.dos.p property 는 FALSE 로 설정되어야 한다.

Device 상태가 RESET 일 때,

- 모든 SVR 컨텐츠가 삭제되고 제조사 기본설정 값으로 리셋된다.
- 기본 제조사 Device 상태는 RESET 이다.
- Vertical resource 는 제조사 기본설정 리셋된다.

- Vertical resource 에는 액세스할 수 없다.
- client 구독자가 dos.p=FALSE 를 관측하면, 액세스 제어 및 크리덴셜 resource (메시지 전달에 필요한)가 해체되는 시점 전에 통지가 전달된다.

- RESET 이 성공적으로 처리되면, /pstat.dos.s 를 RFOTM 으로 설정함으로써 SRM 이 RFOTM 으로 전환된다.

Device 상태가 RFOTM 일 때:

- Vertical Resource 에는 액세스할 수 없다.
- OTM 이 성공적으로 완료되기 전에, /oic/sec/doxm Resource 의 deviceid Property 는 무작위로 추출된 UUID 값으로 설정되어야 한다.
- OTM 이 성공적으로 완료되기 전에, /pstat.dos.s Property 는 비 인가된 요청자에 대해 읽기 전용이다.
- OTM 이 성공적으로 완료되면, /pstat.dos.s Property 는 인가된 요청자에 대해 읽기-쓰기로 된다.
- 절충된 Device 소유자 크리덴셜은 OBT 가 Device 상태를 RFPRO 로의 전환으로 인도하기 위한 인증된 세션을 생성하는데 사용된다.
- <tbd=60> 초 후에 OTM 이 완료되었을 때 인증된 세션을 구성할 수 없으면, SRM 은 OTM 실패를 선언하고 RESET (/pstat.dos.s=RESET)으로 전환한다. (주: /doxm.owned 가 TRUE 로 설정되면 소유권 이전이 완료된 것으로 간주한다. Device 상태는 초기 프로비저닝을 완료하기 위해 RFOTM 을 유지하는 경우가 있다.)

Device 상태가 RFPRO 일 때:

- /pstat.dos.s Property 는 비 인가된 요청자에 대해 읽기 전용이고 인가된 요청자에 대해 읽기-쓰기이다.
- Vertical Resource 에는 액세스할 수 없다.
- OCF Server 는 vertical Resource 를 재 생성할 수 있다.
- 인가된 Client 는 RFNOP 에서의 정상적인 기능을 위해 필요에 따라 SVR 을 프로비저닝할 수 있다.

- 인가된 Client 는 어떤 것을 재 프로비저닝할지 결정하기 위해 SVR 에 대해 연속성 점검을 수행할 수 있다.
- SVR 을 성공적으로 프로비저닝하지 못하면 RESET 으로의 상태 전환의 원인이 될 수 있다. 예를 들어, Device 가 이미 SRESET 으로부터 전환되었지만 연속성 점검을 계속해서 실패할 때.
- 인가된 Client 는 /pstat.dos.s=RFNOP 으로 설정한다.

Device 상태가 RFNOP 일 때:

- /pstat.dos.s Property 는 비 인가된 요청자에 대해 읽기 전용이고 인가된 요청자에 대해 읽기-쓰기이다.
- Vertical resource, SVR, 및 core Resource 는 정상적인 액세스 처리에 따라 액세스할 수 있다.
- 인가되면 RFPRO 로 전환할 수 있다. Device 소유자만 SRESET 또는 RESET 으로 전환할 수 있다.

Device 상태가 SRESET 일 때:

- Vertical Resource 에는 액세스할 수 없다. vertical Resource 의 무결성이 의심되지만 SRM 은 이에 대한 액세스 또는 참조를 시도하지 않는다.
- SVR 의 무결성이 보장되지 않지만 일부 SVR Property 에의 액세스가 필요하다. 여기에는 /doxm.devowner, /cred[<devowner>], 및 /pstat.dos 가 포함된다.
- Device 소유자를 식별하고 인가하기 위한 인증서만으로 최소한의 /cred 및 /doxm resource 를 재 생성해서 SRESET 에 대한 Device 소유자 제어를 가능하게 할 수 있다. SRM 이 이러한 Resource 를 구성하지 못하면 RESET 상태로 전환한다.
- 인가된 Client 는 SVR 연속성 점검을 수행한다. 호출자는 RFPRO 에서의 연속적인 프로비저닝을 위해 사용 가능하도록 또는 RFNOP 에서의 정상적인 기능을 위해 필요에 따라 SVR 을 프로비저닝할 수 있다.
- 인가된 Device 소유자는 /pstat.dos.s 를 RFPRO 또는 RFNOP 로 설정함으로써 RESET 상태로 전환되는 것을 피할 수 있다.
- SVR 상의 ACL 은 무효한 것으로 추정한다. 액세스 인가는 Device 소유자 권리에 따라 부여된다.
- SRM 은 Client 지향 동작 모드를 주장한다 (예: /pstat.om=CLIENT_DIRECTED).

프로비저닝 모드 타입은 다양한 Device 프로비저닝 모드를 나열하는 16 비트 마스크이다. 본 문서에서는 특정 값을 선택하지 않고 프로비저닝 모드의 인스턴스를 참조하기 위해 "{ProvisioningMode}"를 사용한다.

Type Name	Type URN	설명
Device Provisioning Mode	urn:oic.sec.dpmtype	Device 프로비저닝 모드는 다양한 프로비저닝 모드를 기술하는 16 비트 비트마스크이다.

3200 **표 53 –oic.sec.dpmtype Property 정의**

값	Device 모드	설명
bx0000,0001 (1)	Reset	제조사 리셋 동작을 가능하게 하는 Device 리셋 모드.
bx0000,0010 (2)	Take Owner	소유권 이전 동작을 가능하게 하는 Device 페어링 모드.
bx0000,0100 (4)	Bootstrap Service	BSS 의 인스턴스화를 가능하게 하는 서비스 프로비저닝 모드. 인가된 개체가 BSS 를 설치할 수 있도록 한다.
bx0000,1000 (8)	Security Management Services	Device 보안 서비스 및 관련된 크리덴셜의 인스턴스화를 가능하게 하는 서비스 프로비저닝 모드.
bx0001,0000 (16)	Provision Credentials	urn:oic.sec.cms 타입의 관리 서비스를 사용해서 쌍 Device 크리덴셜의 인스턴스화를 가능하게 하는 크리덴셜 프로비저닝 모드.
bx0010,0000 (32)	Provision ACLs	urn:oic.sec. ams 타입의 관리 서비스를 사용해서 Device ACL 의 인스턴스화를 가능하게 하는 ACL 프로비저닝 모드.
bx0100,0000 (64)	Initiate Software Version Validation	요청된/보류된 소프트웨어 버전 확인 (1) 완료된 소프트웨어 버전 확인 (0)
bx1000,0000 (128)	Initiate Secure Software Update	요청된/보류된 보안 소프트웨어 업데이트 (1) 완료된 보안 소프트웨어 업데이트 (0)

3201 **표 54 – oic.sec.dpmtype Property (하위 바이트)의 값 정의**

값	Device 모드	설명

bx0000,0000 – bx1111,1111	<Reserved>	나중의 사용을 위해 예약

3202 **표 55 – oic.sec.dpmtype Property (상위 바이트)의 값 정의**

3203 *프로비저닝 동작 모드* 타입은 다양한 프로비저닝 동작 모드를 열거하는 8 비트 마스크이다.

3204

Type Name	Type URN	설명
Device Provisioning OperationMode	urn:oic.sec.pomtype	Device 프로비저닝 동작 모드는 다양한 프로비저닝 동작 모드를 기술하는 8 비트 비트마스크이다.

3205 **표 56 – oic.sec.pomtype Property 정의**

값	동작 모드	설명
bx0000,0001 (1)	Server-directed utilizing multiple provisioning services	프로비저닝 관련 서비스가 서로 다른 Device 에 위치한다. 따라서, 프로비저닝된 Device 는 각 서비스에 대해 복수의 DTLS 세션을 구성해야 한다. 비트 0 가 FALSE 일 때 이 조건이 존재한다.
bx0000,0010 (2)	Server-directed utilizing a single provisioning service	모든 프로비저닝 관련 서비스가 동일한 Device 내에 위치한다. 따라서, 프로비저닝된 Device 는 프로비저닝 서비스에 따라 복수의 DTLS 세션을 구성하지 않고 오직 하나의 DTLS 세션만 구성한다. 비트 0 가 TRUE 일 때 이 조건이 존재한다.
bx0000,0100 (4)	Client-directed provisioning	Device 가 Device 의 프로비저닝 동작에 대해 프로비저닝 서비스 제어를 지원한다. 비트 1 이 TRUE 일 때 이 조건이 존재한다. 이 비트가 FALSE 이면 이 Device 는 프로비저닝 단계를 제어한다.
bx0000,1000(8) – bx1000,0000(128)	<Reserved>	나중의 사용을 위해 예약
bx1111,11xx	<Reserved>	나중의 사용을 위해 예약

3206 **표 57 – oic.sec.pomtype Property 의 값 정의**

3207 ### 13.8 Certificate 서명 요청 Resource

3208 /oic/sec/csr Resource 는 Device 가 원하는 아이덴티티, 인증할 공개 키, 및 RFC 2986 PKCS#10
3209 Certification Request 형식의 대응하는 개인 키를 소유하고 있는 증거를 제공하는데 사용된다.
3210 Device 가 인증서를 지원하면 (/oic/sec/doxm 의 sct property 가 0x8 비트 위치에 1 을 갖는다),
3211 Device 는 /oic/sec/csr resource 를 갖는다.

고정 URI	Resource Type Title	Resource Type ID ("rt" 값)	Interface	설명	관련 기능 연동
/oic/sec/csr	Certificate Signing Request	urn:oic.r.csr	baseline	CSR resource 는 Device 의 공개 키에 대한 Certificate Signing Request 를 포함한다.	구성

3212 **표 58 – oic.r.csr Resource 정의**

Property Title	Property Name	Value Type	액세스 모드	필수	설명
Certificate Signing Request	csr	스트링	R	예	인코딩 Property 에 따라 인코딩된 서명된 CSR 을 포함한다.
Encoding	encoding	스트링	R	예	csr Property 에 포함된 데이터의 인코딩 형식을 지정하는 스트링. "oic.sec.encoding.pem" – PEM 인코딩된 인증서 서명 요구를 위한 인코딩 "oic.sec.encoding.der" – DER 인코딩된 인증서 서명 요구를 위한 인코딩

3213 **표 59 – oic.r.csr Resource 의 Property**

3214 Device 는 사용할 공개 키를 선택하고, 이 목적을 위해 선택적으로 신규 키 쌍을 생성할 수 있다.

3215

3216 CSR 에서, Subject Name 의 Common Name 요소는 "uuid:X" 형식의 스트링을 포함한다. 여기서,
3217 X 는 RFC 4122 에 의해 정의된 형식의 Device 의 요청된 UUID 이다. Common Name 및 그 밖의
3218 Subject Name 의 요소는 그 밖의 데이터를 포함할 수 있다. Device 가 Common Name 요소에 추가
3219 정보를 포함하고자 할 때는, 공백, 콤마, 또는 세미콜론으로 UUID 필드와 구분한다.

3220

3221 Device 가 사용할 사전 프로비저닝된 키 쌍을 갖고 있지 않지만 신규 키 쌍을 생성하는 기능이
3222 있으면, Device 는 이 resource 의 RETRIEVE 결과로 키 쌍의 생성을 시작할 수 있다. Device 가 키
3223 쌍의 생성에 필요한 시간으로 인해 RETRIEVE 요청에 즉시 응답하지 못하는 경우, Device 는

"operation pending" 에러를 리턴한다. 이는 Client 에게 Device 가 아직 응답할 준비가 되어 있지 않지만 나중에 응답할 수 있음을 알려준다. 이 경우, Client 는 잠시 후에 요청을 재 시도한다.

13.9 Role resource

role resource 는, 섹션 10.3.1 에 기술된 바와 같이, role 인증서로 주장된 role 을 보유한다. 주장된 role 은 관련된 공개 키, 즉 role 인증서 내의 공개 키를 갖는다. Client 는 during (D)TLS 세션 구성 시에 인증에 사용된 인증서의 공개 키와 관련된 role 에만 액세스할 수 있다. role resource 는 (D)TLS 세션 상태의 연장으로 보아야 한다. role 인증서의 검증 방법에 관해서는 섹션 10.3.1 을 참조하기 바란다.

role resource 는 이미 생성되어 있지 않으면 client 와의 보안 (D)TLS 세션 구성 시에 server 에 의해 생성된다. server 는 최소한 (D)TLS 세션이 존재하는 동안 role resource 를 유지한다. server 는 최소한 인증서가 만료되거나 (D)TLS 세션이 종료할 때까지, 어느 쪽이든 빠른 쪽으로, role resource 내에 각 인증서를 유지한다. server 는, 사용 시점에서의 인증서의 유효 기간을 확인하기 위한 섹션 10.3 및 10.3.1 이 상시 적용되기는 하지만, (D)TLS 세션 또는 인증서의 유효 기간의 길이를 초과해서 role resource 및 그 컨텐츠를 유지할 수 있다. server 는 role resource 의 컨텐츠를 정기적으로 검사해서 resource 제약을 토대로 결정하는 정책에 따라 컨텐츠를 정리해야 한다. 예를 들어, 만료된 인증서나 일정 기간 동안 접촉이 없는 client 로부터의 인증서가 정리 대상이 될 수 있다.

위에 언급한 대로, resource 는 (D)TLS 세션의 구성 시에 server 에 의해 암묵적으로 생성된다. 구체적으로, Role Resource 에 대한 RETRIEVE, UPDATE, 및 DELETE 동작은 다음과 같이 이루어진다. 나열되지 않은 동작은 구현 특정적이며 안정적이지 않다. 단, 여기서의 기술은 사설적인 것으로, 규범적이고 공적인 동작 기술은 RAML 에서 제공한다.

1. Retrieve 는 모든 이전에 주장된 client 의 공개 키에 관련된 role 을 리턴한다. server 에게 공개 키는 보안 채널 정보의 일환으로 상시 사용 가능하다. 질의 파라미터를 갖는 Retrieve 는 지원되지 않는다.

2. Update 는 "roles" array property 를 포함하고, 이 배열의 개별 role 은 resource 에 추가된다. 이것은 또한 client 의 공개 키에 범위가 한정된다. "role" 또는 "authority" property 가 다르면 두 개의 role 은 서로 다르다.

3. Delete 는 client 의 공개 키에 대한 전체 "role" 배열을 삭제한다.

고정 URI	Resource Type Title	Resource Type ID ("rt" 값)	Interface	설명	관련 기능 연동
/oic/sec/roles	Roles	urn:oic.r.roles	baseline	이 server 에 대해 이전에 주장되었던 role 을 포함하는 Resource	보안

3255 **표 60 – oic.r.roles Resource 정의**

Property Title	Property Name	Value Type	Value Rule	액세스 모드	필수	설명
Roles	roles	oic.sec.cred	배열	RW	예	이 server 에 대해 이전에 주장되었던 role 의 목록

3256 **표 61 – oic.r.roles Resource 의 Property**

3257 ### 13.10 Security Virtual Resources (SVR) 및 액세스 정책

3258 SVR 은 Device 의 보안 관련 Property 를 노출시킨다.

3259 이러한 SVR 에 대한 비 인가 (익명) Client 의 액세스 요청 (RETRIEVE, UPDATE, DELETE 등)을
3260 승인하면 프라이버시 또는 보안 문제를 일으킬 수 있다.

3261 예를 들어, Device onboarding State 가 RFOTM 이면 oic.r.doxm Resource 에 대한 익명의
3262 요청자의 요청을 승인할 필요가 있으므로 OBT 에 의해 Device 가 발견되어 온보딩될 수 있다.
3263 결과적으로 프라이버시 보호를 위해, oic.r.doxm Resource 에 대한 익명의 요청자의 요청은
3264 거절하는 것이 바람직할 수 있다.

3265 ### 13.11 SVR, Discoverability, 및 Endpoint

3266 모든 구현된 SVR 은 "발견 가능"하다 (다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로
3267 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가
3268 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

3269

3270 OIC Core Specification, Policy Parameter 섹션 7.8.2.1.2 참조).

3271

3272 모든 구현된 발견 가능한 SVR 은 Secure Endpoint (예: CoAPS)를 노출시킨다 (다음의 문헌은, 일부
3273 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의

경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification, Endpoint 챕터 10 참조).

/doxm Resource 는 RFOTM 에서 Unsecure Endpoint (예: CoAP)를 노출시킨다 (다음의 문헌은, 일부 또는 전부가 본 문서에서 규범적으로 인용되며 적용에 있어서 필수적이다. 날짜가 표기된 문헌의 경우에는 인용된 판만 적용된다. 날짜가 표기되지 않은 문헌의 경우에는 참조된 문헌의 최신판(보정 내용 포함)이 적용된다.

OIC Core Specification, Endpoint 챕터 10 참조).

13.12 Core 및 SVR 을 위한 개인 보호 고려 사항

고유 식별자는 잠재적으로 추적 메커니즘으로 사용될 수 있으므로 프라이버시를 위한 고려 사항 중 하나이다. 이는 다음과 같은 Resource 및 Property 를 포함한다.

- 'di' 및 'piid' Property 를 포함하는 /d Resource
- 'pi' Property 를 포함하는/p Resource
- 'deviceuuid' Property 를 포함하는/doxm Resource

모든 식별자는 Device 의 수명이 다 할 때까지 익명의 요청자가 볼 수 있는 고유한 값이다. 이는 악의적인 의도를 가진 개체를 포함하는 모든 Client Device 가 특정 Platform 및 Device 와 관련된 활동 로그를 구축하는데 유용한 식별자를 안정적으로 취득할 수 있음을 의미한다.

Device 의 프라이버시 보호를 위한 전략에는 다음과 같은 두 가지가 있다.

1. 고유 식별자를 포함하는 Resource 에의 읽기 액세스를 제한하는 ACL 정책을 적용한다.
2. 추적에 사용할 수 없도록 식별자의 지속성을 제한한다.

프라이버시 공격에의 노출을 줄이도록 두 가지 기법을 함께 효과적으로 사용할 수 있다.

1. Platform/Device 제조사는 익명의 요청자의 고유 식별자에의 액세스를 제한하는 기본 ACL 정책을 지정해야 한다. 네트워크 관리자는 프라이버시 위협을 보이지 않는 인증된 Device 에 대해 액세스를 허용하도록 ACL 정책을 수정해야 한다.

2. Server 는 /doxm Resource 의 'owned' Property 가 FALSE 이고 복수의 소유권 이전에 걸쳐서 적용될 때 임시적인 비 반복 Device ID 를 제공한다. 임시 식별자는 지속적인 식별자로부터 분리되어 상관성을 갖지 않는다.

    a. 지속적인 식별자로부터 분리된다 (즉, 링크되지 않는다)

3. Generated by a function that is 사전 이미지에 내성을 갖고, 두 번째 사전 이미지에 내성을 갖고, 충돌에 내성을 갖는 기능에 의해 생성된다.

배치되고자 하는 신규 Device 는 온보딩 프로세스의 개시에 사용된 식별자의 OBT 에 통보할 필요가 있다. 그러나, 공격자도 이 값을 취득해서 Device 의 수명이 다 할 때까지 Device 의 추적에 사용할 수 있다. 이러한 프라이버시 위협을 해소하기 위해, Server 는 'deviceowneruuid'가 nil UUID 일 때 unauthenticate /oic/res 요청에 대해 임시 'deviceuuid'를 제공한다. Server 는 Device 상태가 RESET 으로 전환될 때 신규 의사 난수 임시 'deviceuuid' 값을 생성한다. 이렇게 함으로써 'deviceuuid' 값을 복수의 소유자에 걸쳐서 추적하는데 사용할 수 없게 된다.

'deviceowneruuid' property 는 RESET 시에 nil UUID 로 초기화되어 RFOTM Device 상태에서 설정될 때까지 유지된다. Device 는 'deviceowneruuid'가 nil UUID 인 동안 /oic/sec/doxm 및 /oic/res Resource 에 대한 RETRIEVE 요청에 대해 일시적인 식별자를 제공한다. 온보딩 유틸리티는 'devowneruuid'를 Server 가 /doxm 및 /res resource 에 대한 RETRIEVE 요청에 대해 지속적인 'deviceuuid'의 리턴을 개시하도록 하는 영이 아닌 UUID 값으로 갱신한다.

온보딩 유틸리티는 인증된 Client 만이 지속적인 'deviceuuid' 값을 취득할 수 있도록 /oic/sec/doxm resource 에의 액세스를 제한하는 ACL 정책을 프로비저닝할 수도 있다. Client 는 Server 의 'deviceuuid'를 포함하는 /oic/sec/cred resource 항목과 함께 프로비저닝됨으로써 비 인증 발견 요청을 방지한다.

/oic/d Resource 의 'di' property 는 'deviceuuid' Property 를 반영한다. 온보딩 유틸리티는 인증된 Client 만이 지속적인 'di' 값을 취득할 수 있도록 /oic/d Resource 에의 액세스를 제한하는 ACL 정책을 프로비저닝해야 한다.

/oic/d Resource 의 'piid' Property 는 'deviceowneruuid'가 nil UUID 값을 가질 때 마찬가지로 일시적이고 변하는 값을 제공해야 한다. server 는 'deviceowneruuid'가 영이 아닌 UUID 값으로 변경됨에 따라 지속적인 값을 제공한다(또는 온보딩 유틸리티가 프로비저닝하도록 한다). /d resource 에 대한 ACL 정책은 'piid'가 익명의 요청자에게 공개되는 것을 방지한다.

/oic/p Resource 의 'pi' Property 는 'deviceowneruuid'가 nil UUID 값을 가질 때 일시적이고 변하는 값을 가지며, 'deviceowneruuid'가 영이 아닌 UUID 값으로 변경되면 지속적인 값으로 변경된다. ACL 정책은 /p resource 의 'pi' property 가 익명의 요청자에게 공개되는 것을 방지한다.

Resource Type	Property title	Property name	Value type	액세스 모드		동작
oic.wk.p	Platform ID	pi	oic.types-schema.uuid	All States	R	Server 가 임시 random UUID 를 생성한다. (지속적인 pi 를 덮어쓰지 않는다)
oic.wk.p	Protocol Independent Identifier	piid	oic.types-schema.uuid	RESET, SRESET RFPRO, RFNOP	R	RESET 상태로 전환될 때 Server 가 임시 random UUID 를 생성해야 한다.
				RFOTM	RW	보안 소유권 이전 세션이 구성된 후에 DOXS 가 지속적인 값을 설정할 수 있다. 그렇지 않으면, Server 가 값을 설정한다.
oic.wk.d	Device Identifier	di	oic.types-schema.uuid	All states	R	모든 Device 상태에서 /d.di 가 /doxm.deviceuuid 에 포함된 값을 반영한다.

표 62 – Core Resource Property 상태

14 Core 연동 패턴 보안

이 섹션은 의도적으로 빈 칸으로 남겨 두었으며 이후의 버전에서 정의된다.

14.1 관찰자

14.2 구독/통지

14.3 그룹

14.4 발행-구독 패턴 및 통지

15 보안 강화 가이드라인/실행 환경 보안

이 섹션은 참고 섹션이다. OCF 의 다양한 TG 는 그들의 프로토콜 및 환경을 위한 보안 고려 사항을 갖는다. 이러한 보안 고려 사항은 OCF 보안 스펙에 규정된 보안 메커니즘을 통해 해결된다. 그러나, 이러한 메커니즘의 효과는 근본적인 하드웨어 및 소프트웨어 Platform 의 보안 견고성에 의존한다. 이 섹션에서는 실행 환경 보안에 요구되는 요소를 정의한다.

15.1 실행 환경 요소

컴퓨팅 Device 내의 실행 환경은 다양한 구성요소를 갖는다. 보안 기능을 견고하게 수행하려면, 이러한 구성요소 각각의 안전을 독립된 차원으로 확보해야 한다. 예를 들어, AES 암호화를 수행하는 CPU 의 구역이 다른 프로세스와 독립적으로 동작하더라도 실행 엔진으로 키를 입력하는 입력 경로가 안전하지 않으면 AES 를 수행하는 실행 환경이 안전하다고 할 수 없다. 실행 환경의 요소로 불리는 다양한 차원을 아래에 나열한다. 보안 실행 환경(secure execution environment: SEE)로서의 자격을 갖추려면, 대응하는 SEE 요소가 안전한 자격을 갖춰야 한다.

- (보안) 저장
- (보안) 실행 엔진
- (신뢰) 입력/출력 경로
- (보안) 타임 소스/클럭
- (임의) 숫자 생성기
- (승인) 암호화 알고리즘
- 하드웨어 조작 (보호)

소프트웨어 보안 실무 (예를 들어, OWASP 에 의해 다루어지는)는 보안 코드의 개발이 오픈 소스 개발 커뮤니티가 동반되는 실무이므로 본 스펙의 범위에 포함되지 않는다. 단, 본 스펙에서는 소프트웨어의 실행에 요구되는 기본적인 Platform 지원을 다룬다. 보안 부팅 및 보안 소프트웨어 업그레이드를 예로 들 수 있다.

위에 나열된 각각이 요소를 다음의 서브섹션에서 설명한다.

15.1.1 보안 저장

보안 저장은 민감하거나 기밀성 데이터 ("민감 데이터 (Sensitive Data)")를 저장하는 물리적 방법을 말한다. 그러한 데이터는 이에 한정되지는 않지만 대칭 또는 비 대칭 개인 키, 인증서 데이터, 네트워크 액세스 크리덴셜, 또는 사용자 개인 정보를 포함한다. *중요한* 민감 데이터 (*Critical* Sensitive Data)는 무결성과 기밀성을 둘 다 유지해야 하는 반면, Sensitive Data 는 무결성을 유지해야 한다.

IoT Device 생산자가 인가되지 않은 Device, 그룹, 또는 개인이 악의적이거나 악의 없는 목적을 위해 Sensitive Data 에 액세스하지 못하도록 타당한 보호를 제공할 것을 강력히 권한다. 뿐만 아니라, Sensitive Data 는 인증 및 암호화에 종종 사용되므로, 의도적이거나 우발적인 변경에 대해 무결성을 유지해야 한다.

Sensitive Data 의 부분적인 목록을 아래에 나열한다.

데이터	완전성 보호	기밀성 보호
Owner PSK (Symmetric Keys)	예	예
Service provisioning keys	예	예
Asymmetric Private Keys	예	예
Certificate Data and Signed Hashes	예	불필요
Public Keys	예	불필요
Access credentials (e.g. SSID, passwords, etc.)	예	예
ECDH/ECDH Dynamic Shared Key	예	예
Root CA Public Keys	예	불필요
Device and Platform IDs	예	불필요

표 63 – Sensitive Data 예

보안 저장을 위한 정확한 보호 방법은 구현 특정적이지만, 통상적으로 하드웨어와 소프트웨어를 조합하는 방법이 사용된다.

15.1.1.1 하드웨어 보안 저장

하드웨어 보안 저장은 대칭 및 비 대칭 개인 키, 액세스 크리덴셜, 및 개인 정보와 같은 critical Sensitive Data 의 저장에 사용하는 것이 좋다. 대부분의 경우 하드웨어 보안 저장은 반도체 기반의 비휘발성 메모리 ("NVRAM")를 포함하고, Critical Sensitive Data 에의 인가되지 않은 액세스를 방지하기 위한 대책을 포함한다.

하드웨어 기반의 보안 저장은 NVRAM 내에 Sensitive Data 를 저장할 뿐 아니라 물리적 및/또는 전자적 공격을 통한 Sensitive Data 의 검색을 방지하는 보호 메커니즘을 제공한다. 그러한 공격 자체를 방지할 필요는 없지만, 시도된 공격으로 인해 인가되지 않은 개체가 성공적으로 Sensitive Data 를 검색을 초래하지 않도록 해야 한다.

보호 메커니즘은, 이에 한정되지는 않지만 다음과 같은 사항을 포함하는, 공격으로부터 Sensitive Data 에의 액세스에 대해 JIL Moderate protection 을 제공해야 한다.

1) 광학적으로 NVRAM 의 내용을 읽기 위한 칩 패키지의 물리적 외피 제거
2) 전자적으로 NVRAM 의 내용을 읽기 위한 외피 제거된 칩 패키지의 물리적 프로빙
3) Critical Sensitive Data 의 비트 패턴을 파악할 목적으로 전압 변동을 모니터하기 위한 전력선 또는 RF 방출의 프로빙
4) 마이크로컨트롤러 내에서 저장되거나 전송 중인 메모리 내용을 읽기 위한 악의적인 소프트웨어 또는 펌웨어의 사용
5) 부적절한 Device 동작 또는 Sensitive Data 의 손실 또는 변경을 유도하는 결함의 삽입

15.1.1.2 소프트웨어 저장

데이터를 암호화하더라도 Sensitive Data 를 저장하는 비 보안 메모리와 소프트웨어에만 의존하는 것은 일반적으로 권장하지 않는다. 인증 및 암호 키와 같은 Critical Sensitive Data 의 저장에는 가능한 한 하드웨어 보안 저장을 사용해야 한다.

휘발성 또는 비휘발성 메모리에 저장된 Sensitive Data 는 섹션 15.1.1.1 에 기술된 방법을 통해 인가되지 않은 개체가 액세스하지 못하도록 허용 가능한 알고리즘을 사용해서 암호화해야 한다.

15.1.1.3 추가 보안 가이드라인 및 가장 좋은 실무

아래는 다양한 형태의 보안 공격으로 인해 Sensitive Data 가 손상되지 않도록 하기 위한 일반적인 실무의 예이다.

1) FIPS Random Number Generator ("RNG") – 인증 시도에 사용되는 RNG 의 난수성 및 엔트로피가 부족하면 보안 강도를 저하시킬 수 있다. 이러한 이유로, 모든 인증 시도에 대해 인증된 잡음원과 함께 FIPS 800-90A 호환 RNG 를 사용하는 것이 좋다

2) 보안 다운로드 및 부팅 – 악의적인 소프트웨어의 로딩 및 실행을 방지하기 위해, 내용과 더불어 바이너리 소스를 인증하는 Secure Download 및 Secure Boot 방법을 사용하는 것이 좋다.

3) 사용되지 않는 알고리즘 – 이에 한정되지는 않지만, 아래의 목록에 포함된 알고리즘은 안전하지 않은 것으로, 보안 관련 기능에 대해 사용하지 않아야 한다.

    a. SHA-1

    b. MD5

    c. RC4

    d. RSA 1024

4) 블록 또는 요소 간의 암호화 전송 – critical Sensitive Data 가 Secure Storage 에 저장되어 있더라도, Secure Storage 외부로 전송할 필요가 있을 때는 데이터를 MCU/MPU 내에서 악의적인 소프트웨어에 의한 도청을 방지하기 위해 데이터를 암호화해야 한다.

15.1.2 보안 실행 엔진

실행 엔진은 암호화 알고리즘 또는 보안 프로토콜 (예: DTLS)과 같은 보안 기능을 처리하는 컴퓨팅 Platform 의 일부이다. 실행 엔진의 안전 확보에는 다음과 같은 사항이 요구된다.

- 인가되지 않은 개체/프로세스로부터 민감 프로세스 실행의 분리. 여기에는 CPU 캐시의 분리 및 신뢰 (암호) 경계의 일부로 간주할 필요가 있는 모든 실행 요소의 분리가 포함된다.

- 실행 엔진에 대한 입출력 데이터 경로의 분리. 암호화 전 또는 복호화 후의 암호화되지 않은 sensitive data 또는 복호화나 서명과 같은 암호화 알고리즘에 사용되는 암호화 키를 예로 들 수 있다. 더 자세한 사항은 신뢰 경로를 참조하기 바란다.

15.1.3 신뢰할 수 있는 입력/출력 경로

신뢰/암호화 경계에 대한 데이터 입출력에 사용되는 경로/포트는 보호되어야 한다. 이는 보안 실행 엔진 및 보안 메모리에 대한 입출력 경로를 포함한다.

경로 보호는 하드웨어 기반 (예: 우선 버스의 사용) 및 소프트웨어 기반 (신뢰할 수 없는 버스에 대한 암호화 사용) 둘 다를 포함할 수 있다.

15.1.4 보안 클럭

많은 보안 기능이 시간에 민감한 크리덴셜에 의존한다. 시간 스탬프 Kerberos 티켓, OAUTH 토큰, X.509 인증서, OSCP 응답, 소프트웨어 업그레이드 등을 예로 들 수 있다. 클럭 소스의 보안 결여는 공격자가 시스템 클럭을 수정해서 검증 메커니즘을 무력화시킬 수 있음을 의미한다. 그러므로, SEE 는 부당 변경으로부터 보호되는 보안 타임 소스를 제공해야 한다. 보안 견고성 관점에서 신뢰성은 정확성과는 다르다. NTP 와 같은 프로토콜은 네트워크로부터 비교적 정확한 타임 소스를 제공하지만 공격자에 대한 대항력은 갖추고 있지 않다. 반면에, 보안 타임 소스는 대응하는 보안 메커니즘의 시간 감도에 따라 초 또는 분 단위로 틀릴 수 있다. 신뢰할 수 있는 소스에 의해 서명되고 로컬 Device 내의 서명 검증이 신뢰할 수 있는 프로세스이면 (예: 보안 부팅으로 지원) 보안 타임 소스는 외부의 것일 수 있다.

15.1.5 공인된 알고리즘

전체 생태계의 보안에 있어서 중요한 요소는 공식적으로 점검되고 상호 검토된 (예: NIST 승인) 암호화 알고리즘의 견고성이다. 암호화 알고리즘이 모호하면 보안을 확보할 수 없다. 상호 운용성과 보안을 둘 다 확보하려면, 일반적으로 받아들여지는 암호화 알고리즘을 사용할 뿐 아니라 공인된 암호화 기능의 목록도 명시적으로 지정해야 한다. 신규 알고리즘은 NIST 승인을 얻고 오래된 알고리즘은 더 이상 사용하지 않으므로, OCF 는 승인된 알고리즘의 목록을 보유하고 있어야 한다. 그 밖의 모든 알고리즘은 (일부에 의해 더 강력하다고 여겨지더라도) 승인되지 않은 것으로 간주해야 한다.

승인된 것으로 간주되는 알고리즘은 다음과 같다.

- Hash functions
- Signature algorithms
- Encryption algorithms
- Key exchange algorithms

- Pseudo Random functions (PRF) used for key derivation

이 목록은 본 스펙 또는 별도의 보안 견고성 규칙 스펙에 포함되며, OCF 내의 모든 보안 스펙에 있어서 반드시 따라야 한다.

15.1.6 하드웨어 조작 방지

하드웨어 조작 방지는 다양한 레벨로 이루어진다. 암호화 모듈에 대한 조작 방지에 관해 FIPS 140-2 용어 (필수 사항은 아님)를 차용한다.

- Production-grade (하위): 이것은 환경 또는 그 밖의 물리적 손상을 방지하기 위해 모듈의 회로 위에 도포하는 등각 코팅을 포함하는 요소를 의미한다. 단, 이것은 물리적인 유지보수 시에 비밀 자료를 영으로 만드는 과정을 필요로 하지는 않는다. 이 정의는 FIPS 140-2 security level 1 으로부터 차용된 것이다.

- Tamper evident/proof (중위): 이것은 Device 가 (커버, 케이스, 또는 밀봉 등을 통해) 물리적 조작의 시도 증거를 보이는 것을 의미한다. 이 정의는 FIPS 140-2 security level 2 로부터 차용된 것이다.

- Tamper resistance (상위): 이것은 통상적으로 모듈 상에서 민감 자료를 영으로 만드는 과정을 포함하는 물리적인 조작에 대한 응답을 가짐을 의미한다. 이 정의는 FIPS 140-2 security level 3 으로부터 차용된 것이다.

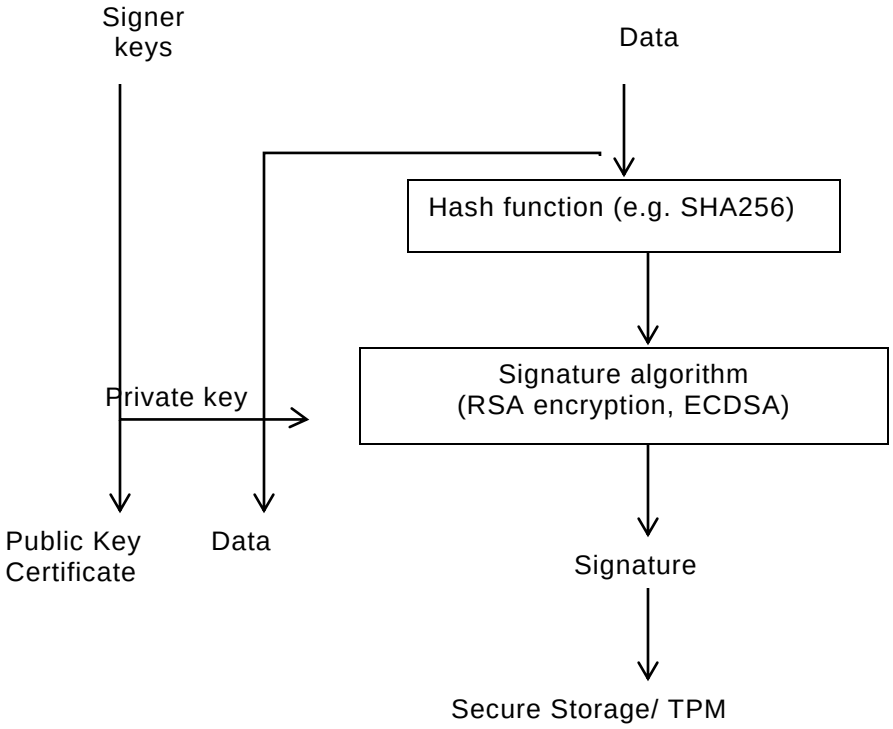
이러한 레벨을 인정하기 위한 정량적인 인증 테스트 사례를 지정하는 것은 쉽지 않다. 컨텐츠 보호 제도는 보통 제조 시에 장착되는 하드웨어 보호를 피하기 위해 사용되는 다양한 툴 (폭넓게 구할 수 있고 전문적인 툴)을 언급한다. 필요한 경우, OCF 가 멤버에 대한 민감 키 자료 (예: PKI)의 배포에 관여되면, OCF 는 그러한 모델을 따를 수 있다.

15.2 보안 부팅

15.2.1 소프트웨어 모듈 인증 개념

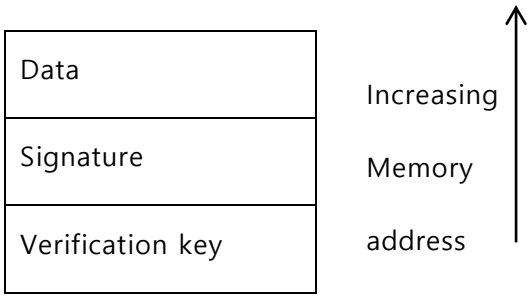
Device 의 모든 요소가 적절하게 동작하고 부당하게 변경되지 않았음을 확실하게 하려면 Device 가 적절하게 부팅되는 것을 확실하게 하는 것이 가장 좋다. 부팅에는 복수의 단계가 있다. 부팅의 최종 결과는 운영체제 상에서 해당하는 드라이버를 통해 메모리, CPU, 및 주변기기를 사용하는 애플리케이션의 실행이다.

일반적인 개념은 각 소프트웨어 모듈이 암호화 무결성 검증이 완료된 후에만 기동되는 것이다. 무결성 검증은 서명한 기관만 액세스할 수 있는 키를 사용해서 해시된 다음에 (예: SHA_1, SHA_256) 암호화 서명 알고리즘 (예: RSA)을 사용해서 서명된 소프트웨어 모듈에 의존한다.



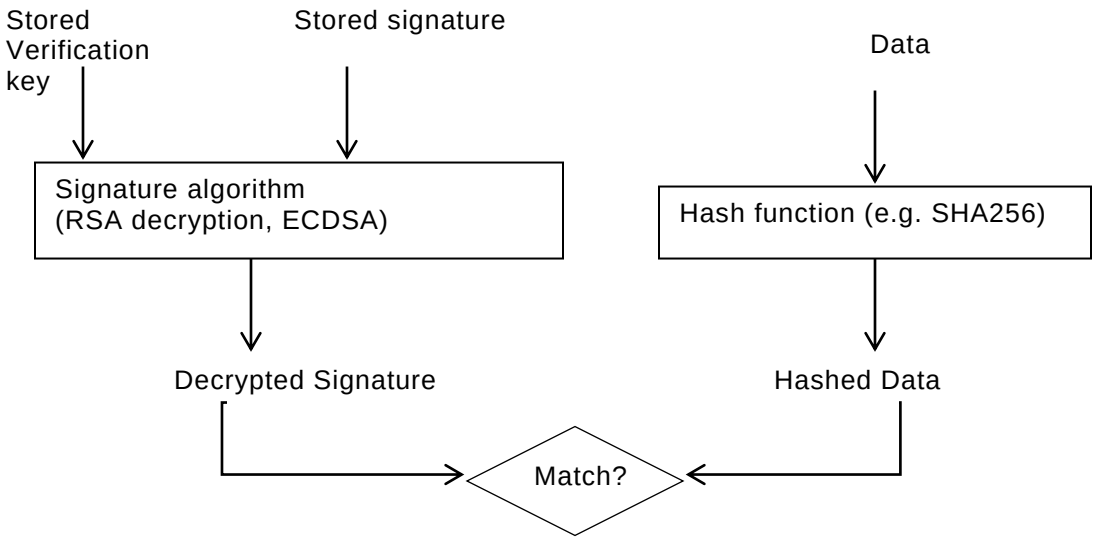
도 40 – 소프트웨어 모듈 인증

서명자의 서명 키 (개인 키)를 사용해서 데이터의 서명이 완료되면, 이후의 검증을 위해 검증 키 (개인 서명 키에 대응하는 공개 키)가 제공된다. 부트로더와 같은 하위 소프트웨어 모듈에 대해서는, 서명 및 검증 키가 1 회 프로그램 가능 메모리 또는 TPM 과 같은 부당 변경 방지 메모리 내에 삽입된다. 애플리케이션 소프트웨어와 같은 상위 소프트웨어 모듈에 대해서는, 통상적으로 signedData 형식이 서명 검증 키 (또는 인증서)뿐 아니라 서명 알고리즘, 해시 알고리즘에 대한 표시를 포함하는 PKCS#7 형식 (IETF CMS RFC)에 따라 서명이 수행된다. 하지만, 보안 부팅 사양은 PKCS#7 형식의 사용을 요구하지 않는다.



도 41 – 검증 소프트웨어 모듈

검증 모듈은 먼저 검증 키 (서명자의 공개 키)를 사용해서 서명을 복호화한다. 또한, 검증 모듈은 데이터의 해시를 산출하고, 복호화된 서명 (오리지널)과 데이터 (실제)의 해시를 비교한다. 두 값이 일치하면, 소프트웨어 모듈은 정품이다.



도 42 – 소프트웨어 모듈 Authenticity

15.2.2 보안 부팅 프로세스

Device 구현에 따라 복수의 부팅 단계가 있다. 통상적으로, PC/ Linux 타입의 환경에서, 첫 번째 단계는 BIOS 코드 (1 단계 부트로더)를 찾아 실행해서 부트 코드가 있는 곳을 찾은 다음에, 부트 코드 (2 단계 부트로더)를 실행하는 것이다. 2 단계 부트로더는 통상적으로 운영체제 (Kernel)을 로드하고 Kernel 코드가 있는 곳으로 실행을 넘기는 프로세스이다. Kernel 이 기동하면 외부 Kernel 모듈 및 드라이버를 로드한다.

보안 부팅 실행 시에는, 부트로더의 실행 전에 각 부트로더의 무결성이 검증되어야 한다. 위에 언급한 바와 같이, 하위 부트로더를 위한 서명 및 검증 키가 통상적으로 조작 방지 메모리 내에 저장되는데 반해, 상위 서명 및 검증 키는 데이터 구조 소프트웨어 내에 내장되어야 (단, 쉽게 액세스 가능하도록 부착되어야) 한다.

15.2.3 견고성 요구 사항

높은 견고성을 가진 보안 부팅 프로세스를 위해서는, 서명 및 해시 알고리즘이 승인된 알고리즘 중 하나이어야 하고, 검증에 사용되는 서명 값 및 키가 보안 저장 내에 저장되어야 하고, 알고리즘이 보안 실행 환경 내에서 실행되어야 하고, 키가 신뢰할 수 있는 경로를 통해 SEE 에 제공되어야 한다.

15.2.3.1 다음 단계

공인된 알고리즘 및 데이터 형식의 목록 작성.

15.3 인증

15.4 소프트웨어 업데이트

15.4.1 개요

Device 의 생명주기는 Device 가 제조사로부터 선적된 시점에서 끝나지 않고, 유통, 판매, 구입, 설치/온보딩, 정상사용, 유지보수, 및 수명 만료 단계까지 남아있게 된다. Device 는 이들 중 어느 단계에서도 갱신을 필요로 할 수 있으며, 통상적으로는 온보딩, 정상사용, 및 유지보수 시에 갱신을 필요로 한다. 소프트웨어는, 이에 한정되지는 않지만, 펌웨어, 운영체제, 네트워킹 스택, 애플리케이션 코드, 드라이버 등을 포함한다.

15.4.2 현재의 차이 인식

다양한 제조사가 다양한 툴 및 전략을 통해 소프트웨어를 갱신한다: 무선 또는 유선 USB 연결, 기존 소프트웨어의 전체 또는 부분 대체, 서명 및 검증된 코드, 전달 패키지의 증명, 코드의 소스 검증, 소프트웨어의 패키지 구조 등.

제조사는 완전한 보안 검토와 초기 아키텍처 구성 후의 주기적인 검토가 지속되는 업계의 기준을 준수하기 위해 자신의 프로세스 및 기술을 검토하는 것이 좋다.

본 스펙은 권장 사항에 따라 Device 에 의해 구현되는 소프트웨어 갱신에 적용될 뿐, 위에 언급한 대체 사유 소프트웨어 갱신 메커니즘과는 아무런 관계가 없다.

15.4.3 소프트웨어 버전 확인

/oic/sec/pstat.tm Property 내의 소프트웨어 버전 확인 개시 (Initiate Software Version Validation) 비트의 설정은 (섹션 13.7 의 표 51 참조) 소프트웨어 버전 확인 프로세스를 개시하는 요구를 나타낸다. 이 프로세스에 의해 Device 는 신뢰할 수 있는 소스에 대해 소프트웨어 (펌웨어, 운영체제, Device 드라이버, 네트워킹 스택 등 포함)를 검증하여 소프트웨어 갱신 프로세스를 시작할 필요가 있는지 파악한다 (아래 참조). 권한을 가진 Client 에 의해 /oic/sec/pstat.tm 의 Initiate Software Version Validation 비트가 1 (TRUE)로 설정되면, Device 가 /oic/sec/pstat.cm 의 Initiate Software Version Validation 비트를 0 으로 설정하고 소프트웨어 버전 확인을 개시한다. Device 가 갱신 가능하다고 판단하는 경우, 사용 가능한 갱신이 있으면 /oic/sec/pstat.cm property 의 Initiate Software Version Validation 비트를 1 (TRUE)로 설정하고 사용 가능한 갱신이 없으면 0 (FALSE)로 설정한다. Software Version Validation 프로세스의 완료를 알리기 위해, Device 는 /oic/sec/pstat.tm Property 의 Initiate Software Version Validation 비트를 다시 0 (FALSE)으로 설정한다. Client 에 의해 /oic/sec/pstat.tm 의 Initiate Software Version Validation 비트가 0 (FALSE)으로 설정되면, 갱신 프로세스에 아무런 영향을 미치지 않는다.

15.4.4 소프트웨어 업데이트

/oic/sec/pstat.tm property 의 보안 소프트웨어 갱신 개시 (Initiate Secure Software Update) 비트의 설정은 (섹션 13.7 의 표 51 참조) 소프트웨어 갱신 프로세스를 개시하는 요구를 나타낸다. 권한을 가진 Client 에 의해 /oic/sec/pstat.tm 의 Initiate Secure Software Update 비트가 1 (TRUE)로 설정되면, Device 가 /oic/sec/pstat.cm 의 Initiate Software Version Validation 비트를 0 으로 설정하고 소프트웨어 갱신 프로세스를 개시한다. 소프트웨어 갱신 프로세스 완료 시에, 소프트웨어가 성공적으로 갱신되었으면 Device 가 /oic/sec/pstat.cm property 의 Initiate Secure Software Update 비트를 1 (TRUE)로 설정하고 갱신이 수행되지 않았으면 0 (FALSE)으로 설정한다. 보안 소프트웨어 갱신 프로세스의 완료를 알리기 위해, Device 는 /oic/sec/pstat.tm Property 의 Initiate Secure Software Update 비트를 다시 0 (FALSE)으로 설정한다. Client 에 의해 /oic/sec/pstat.tm 의 Initiate Secure Software Update 비트가 0 (FALSE)으로 설정되면, 갱신 프로세스에 아무런 영향을 미치지 않는다.

15.4.5 권장 사용

/oic/sec/pstat.tm 의 Initiate Secure Software Update 비트는 Initiate Software Version Validation 확인이 완료된 후에 Client 에 의해서만 설정되어야 한다.

Device 소프트웨어 갱신 프로세스는 Device Operational State (/oic/sec/pstat.dos)에 영향을 미치는 상태 변경을 수반할 수 있다. 갱신되는 Device 중에서 해당하는 Device 는 /oic/sec/pstat.dos 를 모니터링해서 갱신이 완료되기 전에 Device 상태에 영향을 미치는 진행 중인 소프트웨어 갱신에 대비해야 한다.

Device 자체가 버전 확인 또는 갱신 프로세스의 시작 또는 완료 시에 pstat.tm 및 pstat.cm Initiate Software Version Validation 및 Secure Software Update 비트를 설정함으로써 소프트웨어 버전 확인/갱신을 자율적으로 개시하거나 확인/갱신이 완료된 것을 나타낼 수 있다. Client 지향 갱신에서는 흔히 있듯이, Client 는 pstat resource 변경을 관측함으로써 자율적인 버전 확인 또는 소프트웨어 갱신이 보류 및/또는 완료된 사실을 통지받을 수 있다.

15.5 Non-OCF Endpoint 상호 운용성

15.7 보안 레벨

Security Level 은 보안 기준을 토대로 Device 를 차별화한다. 이러한 차별화의 필요성은 산업과 헬스케어와 같은 다양한 vertical 로부터의 요구에 근거한 것으로 스마트홈까지 이어질 수 있다. 이 차별화는 Device 분류 (예: RFC7228)와는 별개의 것이다.

이러한 보안 차별화의 범주는, 이에 한정되지는 않지만, 다음과 같은 사항을 포함한다.

1. 보안 강화
2. 아이덴티티 증명
3. 인증서/신뢰
4. 온보딩 기술
5. 규정 준수
    a. 저장 데이터
    b. 전송 데이터
6. Cipher suite – 암호화 알고리즘 & 커브
7. 키 길이
8. 보안 부팅/업데이트

향후 보안 레벨은 상호 운용성을 정의하는데 사용될 수 있다.

다음은 Security 스펙 1.1 에 적용된다.
현재 스펙은 Security Level 0 을 초과하는 레벨은 정의하고 있지 않다. 모든 Device 는 Level 0 으로 지정된다. 추가적인 레벨은 향후 버전에서 정의될 수 있다.

다음과 같은 점에 주의를 요한다.

- 주어진 보안 레벨의 정의는 스펙의 버전에 관계 없이 유지된다.

- 주어진 레벨을 만족하는 Device 는 상위 레벨로 업그레이드할 수 있거나 할 수 없다.

- 상위 레벨의 요구 조건을 충족하면 Device 는 상위 레벨로 재 분류될 수 있다 (예를 들어, Device 가 1.1 버전의 스펙에 맞춰 제작되고 다음 스펙 버전이 security level 1 을 정의하는 경우, 레벨 1 요구 사항을 만족하면 Device 가 레벨 1 으로 평가되어 분류된다).

- 보안 레벨은 최종 사용자가 볼 수 있어야 한다.

16 Appendix A: 액세스 제어 예

16.1 OCF ACL Resource 예

Server 는 모든 호스트된 Resource 가 액세스를 요청하는 Client 에 의한 액세스를 인가한 것을 확인할 필요가 있다. /oic/sec/acl2 Resource 가 Resource 호스트 상에 함께 배치되어 Resource 요청 처리가 안전하고 효율적으로 적용된다. 이 예는 Server 에 대해 액세스 정책 예를 시행하기 위해 /oic/sec/acl2 Resource 가 어떻게 구성될 수 있는지를 보여준다.

```
{
    "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"
}
```

16.2 Access Manager Service 예

AMS 는 액세스 정책의 관리 집중화를 위해 사용되어야 하지만, 지명된 Resource 가 액세스될 때마다 Server 가 AMS 에의 연결을 개방할 것을 요구한다. 이 예는 이 목적을 달성하기 위해 어떻게 /oic/sec/amacl Resource 가 구성되어야 하는지를 보여준다.

```
{
  "resources": [
     // If the {Subject} wants to access the /oic/sh/light/1 Resource at host1 and an Amacl was
     // supplied then use the ~~{1}~~ sacl validation credential to enforce access.
     {"href": /oic/sh/light/1},
     // If the {Subject} wants to access the /oma/3 Resource at host2 and an AM sacl was
     // supplied  then use the ~~{1}~~ sacl validation credential to enforce access.
     {"href": "/oma/3"},
     // If the {Subject} wants to access any local Resource and an Amacl was supplied then use
     // the ~~{1}~~ sacl validation credential to enforce access.
     {"wc": "*"}]
}
```


17 Appendix B: 실행 환경 보안 프로파일

IoT vertical 및 Device 의 기능이 균일하지 않은 점을 고려하면, 모든 IOT 애플리케이션 및 서비스에 대응하여 두루 적용되도록 만든 보안 견고성 요구 사항은 OCF 의 요구를 만족시키지 않으므로, 가변 견고성 (신뢰성), 비용, 및 복잡성의 보안 프로파일이 정의되어야 한다. 공급자의 대형 생태계를 다루기 위해 프로파일은 요구 사항으로서만 정의될 수 있으며, 그러한 요구 사항을 만족하는 정확한 솔루션은 공급자의 개방 또는 사유 구현에 특정적이므로, 대부분의 경우 본 문서의 범위에 포함되지 않는다.

Device 분류가 IETF RFC 7228 (구속된 노드 네트워크용 Terminology) 방법론에 따르는 OCF 스펙의 나머지와 맞추려면, 보안 프로파일의 수를 최대 3 으로 제한한다. 하지만, 세 클래스 각각에 대한 OCF 기능 기준은 IETF 보다 현재 IoT 칩 시장에 더 적합하게 될 것이다.

클래스 0 에서의 resource 의 극도로 낮은 레벨을 고려하면, 클래스 0 Device 는 보안 기능이 없거나 환경 (예: 사람의 가용성) 인자에 의존해서 보안 기능을 수행하는 쉽게 뚫을 수 있는 보안 기능을 갖는 것으로 추정할 수 있다. 이것은 클래스 0 에 SEE 가 없는 것으로 해석할 수 있다.

Platform 클래스	SEE	견고성 레벨
0	아니오	해당 없음
1	예	저
2	예	고

표 64 – OCF 보안 프로파일

기술 참조: 이 분석에서 위와 같은 Platform 분류에 보안 코프로세서 또는 그 밖의 분류 기준을 높이는 하드웨어 보안 기능 (즉, CPU 속도, 메모리, 저장 장치)의 가능성은 고려되지 않았다.

18 Appendix C: RAML 정의

Resource Name	Resource Type	섹션
Access Control List	oic.r.acl	A.1
Access Control List 2	oic.r.acl2	A.2
Managed Access Control List	oic.r.amacl	A.3
Signed Access Control List	oic.r.sacl	A.4
Device Ownership Transfer	oic.r.doxm	A.5
Device Provisioning Status	oic.r.pstat	A.6
Credential	oic.r.cred	A.7
Certificate Signing Request	oic.r.csr	A.8
Roles	oic.r.roles	A.9
Certificate Revocation List	oic.r.crl	A.10

표 65 – OCF SVR RAML

A.1 OICSecurityAclResource

A.1.1 개요

이 resource 는 로컬 액세스 제어 목록을 규정한다.

A.1.2 URI 예

/oic/sec/acl

A.1.3 Resource Type

A.1.4 RAML 정의

```
#%RAML 0.8
title: OICSecurityAclResource
version: v1.1-20161213
```



```
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.baseline"]
 - ace-filtered :
     queryParameters:
       subjectuuid:

/oic/sec/acl:
  description: |
    This resource specifies the local access control list.

  is : ['interface']
  get:
    description: |
      Retrieves the ACL entries.
      When used without query parameters, all the ACE entries are returned.
      When used with a subjectuuid, only the ACEs with the specified
      subjectuuid are returned
      If subjectuuid and resources are specified,
      only the ACEs with the specified subjectuuid and resource hrefs are
      returned.

    is : ['ace-filtered']
    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.acl.json#",
```



```
  "title": "Access Control List information",
  "definitions": {
    "oic.r.acl": {
      "type": "object",
      "properties": {
        "aclist": {
          "type": "object",
          "description": "Subject-based Access Control Entries in the ACL resource",
          "properties": {
            "aces": {
              "type": "array",
              "items": {
                "$ref": "oic.sec.ace.json#/definitions/oic.sec.ace"
              }
            }
          },
          "required": [ "aces" ]
        },
        "rowneruuid": {
          "description": "The value identifies the unique resource owner",
          "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
        }
      }
    }
  },
  "type": "object",
  "allOf": [
    { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
    { "$ref": "#/definitions/oic.r.acl" }
  ],
  "required": [ "aclist", "rowneruuid" ]
}
```

example: /

```
{
  "aclist": {
```



```
                "aces": [
                  {
                    "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
                    "resources": [
                      {
                        "href": "coaps://IP-ADDR/temp",
                        "rel": "some-rel",
                        "rt": ["oic.r.temperature"],
                        "if": ["oic.if.a"]
                      },
                      {
                        "href": "coaps://IP-ADDR/temp",
                        "rel": "some-rel",
                        "rt": ["oic.r.temperature"],
                        "if": ["oic.if.s"]
                      }
                    ],
                    "permission": 31,
                    "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"
            }
```

400:


```
3820          description: |
3821            The request is invalid.
3822
3823      post:
3824        description: |
3825          Updates the ACL resource with the provided values
3826          ACEs provided
3827          in the update not currently in the ACL are added
3828          ACEs that already
3829          exist in the ACL are ignored.
3830          Note that for the purposes of update, equivalency is determined
3831          by comparing the ACE subjectuuid, permission, string comparisons
3832          of all validity elements, and string comparisons of all resource
3833          hrefs.
3834
3835        body:
3836          application/json:
3837            schema: |
3838              {
3839                "$schema": "http://json-schema.org/draft-04/schema#",
3840                "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.acl.json#",
3841                "title": "Access Control List information",
3842                "definitions": {
3843                  "oic.r.acl": {
3844                    "type": "object",
3845                    "properties": {
3846                      "aclist": {
3847                        "type": "object",
3848                        "description": "Subject-based Access Control Entries in the ACL resource",
3849                        "properties": {
3850                          "aces": {
3851                            "type": "array",
3852                            "items": {
3853                              "$ref": "oic.sec.ace.json#/definitions/oic.sec.ace"
3854                            }
3855                          }
```



```
                },
                "required": [ "aces" ]
              },
              "rowneruuid": {
                "description": "The value identifies the unique resource owner",
                "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
              }
            }
          }
        },
        "type": "object",
        "allOf": [
          { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
          { "$ref": "#/definitions/oic.r.acl" }
        ],
        "required": [ "aclist", "rowneruuid" ]
      }
```

example: /

```
      {
        "aclist": {
          "aces": [
            {
              "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
              "resources": [
                {
                  "href": "coaps://IP-ADDR/temp",
                  "rel": "some-rel",
                  "rt": ["oic.r.temperature"],
                  "if": ["oic.if.a"]
                },
                {
                  "href": "coaps://IP-ADDR/temp",
                  "rel": "some-rel",
                  "rt": ["oic.r.temperature"],
                  "if": ["oic.if.s"]
```



```
                    }
                  ],
                  "permission": 31,
                  "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"
          }

    responses :
      400:
        description: |
          The request is invalid.

      201:
        description: |
          The ACL entry/entries is/are created.

      204:
        description: |
          The ACL entry/entries is/are updated.

  delete:
```



```
  description: |
    Deletes ACL entries.
    When DELETE is used without query parameters, all the ACE entries are deleted.
    When DELETE is used with a subjectuuid, only the ACEs with the specified
    subjectuuid are deleted
    If subjectuuid and resources are specified,
    only the ACEs with the specified subjectuuid and resource hrefs are
    deleted.

  is : ['ace-filtered']
  responses :
    200:
      description: |
        The matching ACEs or the entire ACL resource has been successfully deleted.

    400:
      description: |
        The request is invalid.
```

A.1.5 Property 정의

Property name	Value type	필수	액세스 모드	설명
rowneruuid	복수 타입: schema 참조	예		이 값은 고유 resource 소유자를 식별한다.
aclist	객체: schema 참조	예		ACL resource 내의 주체 기반 Access Control Entry
aces (aclist)	배열: schema 참조	예		

A.1.6 CRUDN 동작

Resource	Create	Read	Update	Delete	Notify
/oic/sec/acl		get	post	delete	

A.2 OICSecurityAcl2Resource

A.2.1 개요

이 resource 는 로컬 액세스 제어 목록을 규정한다.

A.2.2 URI 예

/oic/sec/acl2

A.2.3 Resource Type

A.2.4 RAML 정의

```
#%RAML 0.8
title: OICSecurityAcl2Resource
version: v1.0-20161214
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.baseline"]
 - ace-filtered :
     queryParameters:
       aceid:

/oic/sec/acl2:
  description: |
    This resource specifies the local access control list.

  is : ['interface']
  get:
    description: |
      Retrieves the ACL data.
      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.
```



```
is : ['ace-filtered']
responses :
  200:
    body:
      application/json:
        schema: |
          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.acl2.json#",
            "title": "Access Control List information",
            "definitions": {
              "oic.r.acl2": {
                "type": "object",
                "properties": {
                  "aclist2": {
                    "type": "array",
                    "description": "Access Control Entries in the ACL resource",
                    "items": {
                      "$ref": "oic.sec.ace2.json#/definitions/oic.sec.ace2"
                    }
                  },
                  "rowneruuid": {
                    "description": "The value identifies the unique resource owner",
                    "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
                  }
                }
              }
            },
            "type": "object",
            "allOf": [
              { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
              { "$ref": "#/definitions/oic.r.acl2" }
            ],
            "required": [ "aclist2", "rowneruuid"]
          }
```


example: /

```
{
  "aclist2": [
    {
      "aceid": 1,
      "subject": {
        "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
        "role": "SOME_STRING"
      },
      "resources": [
        {
          "href": "/light",
          "rt": ["oic.r.light"],
          "if": ["oic.if.baseline", "oic.if.a"]
        },
        {
          "href": "/door",
          "rt": ["oic.r.door"],
          "if": ["oic.if.baseline", "oic.if.a"]
        }
      ],
      "permission": 24
    },
    {
      "aceid": 2,
      "subject": {
        "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
      },
      "resources": [
        {
          "href": "/light",
          "rt": ["oic.r.light"],
          "if": ["oic.if.baseline", "oic.if.a"]
        },
        {
          "href": "/door",
          "rt": ["oic.r.door"],
```



```
                        "if": ["oic.if.baseline", "oic.if.a"]
                      }
                    ],
                    "permission": 24
                    },
                    {
                      "aceid": 3,
                      "subject": {"conntype": "anon-clear"},
                      "resources": [
                        {
                          "href": "/light",
                          "rt": ["oic.r.light"],
                          "if": ["oic.if.baseline", "oic.if.a"]
                        },
                        {
                          "href": "/door",
                          "rt": ["oic.r.door"],
                          "if": ["oic.if.baseline", "oic.if.a"]
                        }
                      ],
                      "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"
                }
```



```
      400:
        description: |
          The request is invalid.

  post:
    description: |
      Updates the ACL resource with the provided ACEs.
      ACEs provided in the update with aceids not currently in the ACL
      resource are added.
      ACEs provided in the update with aceid(s) already in the ACL completely
      replace the ACE(s) in the ACL resource.
      ACEs provided in the update without aceid properties are added and
      assigned unique aceids in the ACL resource.

    body:
      application/json:
        schema: |
          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.acl2.json#",
            "title": "Access Control List information",
            "definitions": {
              "oic.r.acl2": {
                "type": "object",
                "properties": {
                  "aclist2": {
                    "type": "array",
                    "description": "Access Control Entries in the ACL resource",
                    "items": {
                      "$ref": "oic.sec.ace2.json#/definitions/oic.sec.ace2"
                    }
                  },
                  "rowneruuid": {
                    "description": "The value identifies the unique resource owner",
                    "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
```

```
                }
              }
            }
          },
          "type": "object",
          "allOf": [
            { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
            { "$ref": "#/definitions/oic.r.acl2" }
          ],
          "required": [ "aclist2", "rowneruuid"]
        }
```

example: /

```
        {
          "aclist2": [
            {
              "aceid": 1,
              "subject": {
                "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
                "role": "SOME_STRING"
              },
              "resources": [
                {
                  "href": "/light",
                  "rt": ["oic.r.light"],
                  "if": ["oic.if.baseline", "oic.if.a"]
                },
                {
                  "href": "/door",
                  "rt": ["oic.r.door"],
                  "if": ["oic.if.baseline", "oic.if.a"]
                }
              ],
              "permission": 24
            },
            {
```



```
            "aceid": 3,
            "subject": {
              "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
            },
            "resources": [
              {
                "href": "/light",
                "rt": ["oic.r.light"],
                "if": ["oic.if.baseline", "oic.if.a"]
              },
              {
                "href": "/door",
                "rt": ["oic.r.door"],
                "if": ["oic.if.baseline", "oic.if.a"]
              }
            ],
            "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.
      When DELETE is used without query parameters, all the ACE entries are deleted.
      When DELETE is used with a query parameter, only the ACEs matching the
      specified parameter are deleted.

    is : ['ace-filtered']
    responses :
      200:
        description: |
          The matching ACEs or the entire ACL resource has been successfully deleted.

      400:
        description: |
          The request is invalid.
```

A.2.5 Property 정의

Property name	Value type	필수	액세스 모드	설명
rowneruuid	복수 타입: schema 참조	예		이 값은 고유 resource 소유자를 식별한다.
aclist2	배열: schema 참조	예		ACL resource 내의 Access Control Entry

A.2.6 CRUDN 동작

Resource	Create	Read	Update	Delete	Notify
/oic/sec/acl2		get	post	delete	

A.2.7 참조된 JSON schema

A.2.8 oic.sec.didtype.json

A.2.9 Property 정의

Property name	Value type	필수	액세스 모드	설명
uuid	복수 타입: schema 참조	예		UUID Device ID

A.2.10 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.didtype.json#",
  "title": "Device Identifier Format type",
  "definitions": {
    "oic.sec.didtype": {
          "type": "object",
          "description": "Device identifier",
          "properties": {
            "uuid": {
              "description": "A UUID Device ID",
              "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
            }
          },
          "required": ["uuid"]
        }
    }
}
```

A.2.11 oic.sec.ace2.json

A.2.12 Property 정의

Property name	Value type	필수	액세스 모드	설명
aceid	정수			ACL 내에서 고유한 ACE 를 식별하기 위한 식별자. 갱신 시에 제공되지

				않으면 Server 는 ACE 를 추가하고 여기에 고유한 값을 할당한다.
subject	복수 타입: schema 참조	yes		이 ACE 가 적용되는 주체로, deviceId, role, 또는 와일드카드이다.
validity	배열: schema 참조			유효성은 시간 패턴 객체의 배열이다.
resources	배열: schema 참조			보안 정책이 적용되는 애플리케이션의 resource 를 참조한다.
rt (resources)	복수 타입: schema 참조			존재하면 rt (resource type)가 매칭될 때만 ACE 가 적용된다.
href (resources)	복수 타입: schema 참조			존재하면 href 가 매칭될 때만 ACE 가 적용된다.
wc (resources)	스트링			와일드카드 매칭 정책
if (resources)	복수 타입: schema 참조			존재하면 if (interface)가 매칭될 때만 ACE 가 적용된다.
permission	정수	yes		CRUDN 권한의 비트마스크 인코딩

A.2.13 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
```



```
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.ace2.json#",
  "title": "Subject-based Access Control Entry (ACE) object definition",
  "definitions": {
    "oic.sec.ace2": {
      "type": "object",
      "properties": {
        "aceid": {
          "type": "integer",
          "minimum": 1,
          "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."
        },
        "resources": {
          "type": "array",
          "description": "References the application's resources to which a security policy applies",
          "items": {
            "type": "object",
            "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",
                "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link/properties/href"
              },
              "rt": {
                "description": "When present, the ACE only applies when the rt (resource type) matches",
                "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link/properties/rt"
              },
              "if": {
                "description": "When present, the ACE only applies when the if (interface) matches",
                "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link/properties/if"
              },
              "wc": {
                "type": "string",
                "enum": [ "+", "-", "*" ],
                "description": "A wildcard matching policy",
                "detail-desc": [  "+ - Matches all discoverable resources",
                                  "- - Matches all non-discoverable resources",
```



```
                                "* - Matches all resources"
                ]
              }
            }
          }
        },
        "permission": {
          "type": "integer",
          "description": "Bitmask encoding of CRUDN permission",
          "$ref": "oic.sec.crudntype.json#/definitions/oic.sec.crudntype/properties/bitmask"
        },
        "subject": {
          "description": "The subject to whom this ace applies, either a deviceId, role, or wildcard",
          "anyOf": [
            {
              "$ref": "oic.sec.didtype.json#/definitions/oic.sec.didtype"
            },
            {
              "$ref": "oic.sec.roletype.json#/definitions/oic.sec.roletype"
            },
            {
              "type": "object",
              "properties": {
                "conntype": {
                  "type": "string",
                  "enum": [ "auth-crypt", "anon-clear" ],
                  "description": "This property allows an ACE to be matched based on the connection or
message type",
                  "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"
                ]
              }
            },
            "required": ["conntype"]
          }
```

```
            ]
          },
          "validity": {
            "type": "array",
            "description": "validity is an array of time-pattern objects",
            "items": {
              "$ref": "oic.sec.time-pattern.json#/definitions/time-pattern"
            }
          }
        },
        "required": [ "permission", "subject" ]
      }
    }
}
```

A.2.14 oic.sec.roletype.json

A.2.15 Property 정의

Property name	Value type	필수	액세스 모드	설명
role	스트링	예		식별되는 role 의 ID.
authority	스트링			식별되는 개체의 기관 요소. NULL <Authority>는 로컬 개체 또는 device 이다.

A.2.16 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.roletype.json#",
  "title": "Security Role Types",
  "definitions": {
    "oic.sec.roletype": {
```



```
      "type": "object",
      "description": "Security role specified as an <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or device.",
      "properties": {
        "authority": {
          "type": "string",
          "description": "The Authority component of the entity being identified. A NULL <Authority> refers to the local entity or device."
        },
        "role": {
          "type": "string",
          "description": "The ID of the role being identified."
        }
      },
      "required": ["role"]
    }
  }
}
```

A.2.17 oic.sec.time-pattern.json

A.2.18 Property 정의

Property name	Value type	필수	액세스 모드	설명
recurrence	배열: schema 참조			RFC5545 Recurrence 를 사용하여 반복 규칙을 표현하는 스트링 배열.
period	스트링	예		RFC5545 Period 를 사용하여 기간을 표현하는 스트링.

A.2.19 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.time-pattern.json#",
```



```
  "title": "RFC5545 time period and recurrence rule(s)",
  "definitions": {
    "time-pattern": {
      "type": "object",
      "description": "The time-pattern contains a period and recurrence expressed in RFC5545 syntax",
      "properties": {
        "period": {
          "type": "string",
          "description": "String represents a period using the RFC5545 Period"
        },
        "recurrence": {
          "type": "array",
          "description": "String array represents a recurrence rule using the RFC5545 Recurrence",
          "items": {
            "type": "string"
          }
        }
      },
      "required": [ "period" ]
    }
  }
}
```

A.2.20 oic.sec.crudntype.json

A.2.21 Property 정의

Property name	Value type	필수	액세스 모드	설명
bitmask	정수	예		권한을 나타내는 인코딩된 비트마스크.

A.2.22 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.crudntype.json#",
  "title": "Permission BitMask",
```



```
  "definitions": {
    "oic.sec.crudntype": {
      "description": "OIC CRUDN types",
      "properties": {
        "bitmask": {
          "type": "integer",
          "minimum": 0,
          "maximum": 31,
          "description": "The encoded bitmask indicating permissions",
          "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" ]
        }
      }
    },
    "type": "object",
    "allOf": [
      { "$ref": "#/definitions/oic.sec.crudntype" }
    ],
    "required": ["bitmask"]
  }
```

A.3 OICSecurityAmaclResource

A.3.1 개요

이 resource 는 AMS 에 의해 관리되는 액세스 권한을 갖는 호스트 resource 를 규정한다.

A.3.2 URI 예

/oic/sec/amacl

A.3.3 Resource Type

A.3.4 RAML 정의

```
#%RAML 0.8
title: OICSecurityAmaclResource
version: v1.0-20150819
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.baseline"]

/oic/sec/amacl:
  description: |
    This resource specifies the host resources with access permission that is managed by an AMS.

  is : ['interface']
  get:
    description: |
      Retrieves the amacl data.

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.amacl.json#",
                "title": "Managed Access Control information",
                "definitions": {
                  "oic.r.amacl": {
                    "type": "object",
```



```
                "properties": {
                  "resources": {
                    "type": "array",
                    "description": "Multiple links to this host's resources",
                    "items": { "$ref": "oic.sec.ace2.json#/properties/resources" }
                  }
                }
              }
            },
            "type": "object",
            "allOf": [
              { "$ref": "#/definitions/oic.r.amacl" }
            ],
            "required": [ "resources" ]
          }

        example: /

          {
            "resources": [
              {
                "href": "/temp",
                "rt": ["oic.r.temperature"],
                "if": ["oic.if.baseline", "oic.if.a"]
              },
              {
                "href": "/temp",
                "rt": ["oic.r.temperature"],
                "if": ["oic.if.baseline", "oic.if.s"]
              }
            ]
          }

    post:

      description: |
        Sets the new amacl data

```



```
body:
  application/json:
    schema: |
      {
        "$schema": "http://json-schema.org/draft-04/schema#",
        "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.amacl.json#",
        "title": "Managed Access Control information",
        "definitions": {
          "oic.r.amacl": {
            "type": "object",
            "properties": {
              "resources": {
                "type": "array",
                "description": "Multiple links to this host's resources",
                "items": { "$ref": "oic.sec.ace2.json#/properties/resources" }
              }
            }
          }
        },
        "type": "object",
        "allOf": [
          { "$ref": "#/definitions/oic.r.amacl" }
        ],
        "required": [ "resources" ]
      }

    example: |
      {
        "resources": [
          {
            "href": "/temp",
            "rt": ["oic.r.temperature"],
            "if": ["oic.if.baseline", "oic.if.a"]
          },
          {
            "href": "/temp",
```



```
                "rt": ["oic.r.temperature"],
                "if": ["oic.if.baseline", "oic.if.s"]
              }
            ]
          }

    responses :
      400:
        description: |
          The request is invalid.

      201:
        description: |
          The AMACL entry is created.

      204:
        description: |
          The AMACL entry is updated.

  put:
    description: |
      Creates the new acl data

    body:
      application/json:
        schema: /
          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.amacl.json#",
            "title": "Managed Access Control information",
            "definitions": {
              "oic.r.amacl": {
                "type": "object",
```



```
                "properties": {
                  "resources": {
                    "type": "array",
                    "description": "Multiple links to this host's resources",
                    "items": { "$ref": "oic.sec.ace2.json#/properties/resources" }
                  }
                }
              }
            },
            "type": "object",
            "allOf": [
              { "$ref": "#/definitions/oic.r.amacl" }
            ],
            "required": [ "resources" ]
          }

        example: /

          {
            "resources": [
              {
                "href": "/temp",
                "rt": ["oic.r.temperature"],
                "if": ["oic.if.baseline", "oic.if.a"]
              },
              {
                "href": "/temp",
                "rt": ["oic.r.temperature"],
                "if": ["oic.if.baseline", "oic.if.s"]
              }
            ]
          }

      responses :
        400:
          description: |
```



```
        The request is invalid.

    201:
      description: |
        The AMACL entry is created.

  delete:
    description: |
      Deletes the amacl data.
      When DELETE is used without query parameters, the entire collection is deleted.
      When DELETE uses the search parameter with "subject", only the matched entry is deleted.

    queryParameters:
      subject:
        type: string
        description: Delete the ACE identified by the string matching the subject value.

        required: false
        example: DELETE /myamacl?subject="de305d54-75b4-431b-adb2-eb6b9e546014"

    responses :
      200:
        description: |
          The ACE instance or the the entire AMACL resource has been successfully deleted.

      400:
        description: |
          The request is invalid.
```

A.3.5 Property 정의

Property name	Value type	필수	액세스 모드	설명

resources	배열: schema 참조	예		호스트의 resource 에의 복수의 링크.

A.3.6 CRUDN 동작

Resource	Create	Read	Update	Delete	Notify
/oic/sec/amacl	put	get	post	delete	

A.4 OICSecuritySignedAclResource

A.4.1 개요

이 resource 는 서명된 ACL 객체를 규정한다.

A.4.2 URI 예

/oic/sec/sacl

A.4.3 Resource Type

A.4.4 RAML 정의

```
#%RAML 0.8
title: OICSecuritySignedAclResource
version: v1.0-20150819
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.baseline"]

/oic/sec/sacl:
  description: |
    This resource specifies a signed ACL object.

  is : ['interface']
  get:
    description: |
```



```
4660        Retrieves the sacl resource data.
4661
4662      responses :
4663        200:
4664          body:
4665            application/json:
4666              schema: /
4667                {
4668                  "$schema": "http://json-schema.org/draft-04/schema#",
4669                  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.sacl.json#",
4670                  "title": "Signed Access Control List information",
4671                  "definitions": {
4672                    "oic.r.sacl": {
4673                      "type": "object",
4674                      "properties": {
4675                        "aclist2":  {
4676                          "type": "array",
4677                          "description": "Access Control Entries in the Acl resource",
4678                          "$ref": "oic.sec.ace2.json#/definitions/oic.sec.ace2"
4679                        },
4680                        "signature":   {
4681                          "type": "object",
4682                          "description": "The signature over the ACL resource",
4683                          "$ref": "oic.sec.sigtype.json#/definitions/oic.sec.sigtype"
4684                        }
4685                      }
4686                    }
4687                  },
4688                  "type": "object",
4689                  "allOf": [
4690                    { "$ref": "#/definitions/oic.r.sacl" }
4691                  ],
4692                  "required": ["aclist2", "signature"],
4693                  "additionalItems": false
4694                }
4695
```


example: /

```
{
  "aclist2": [
    {
      "aceid": 1,
      "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
      "resources": [
        {
          "href": "/temp",
          "rt": ["oic.r.temperature"],
          "if": ["oic.if.baseline", "oic.if.a"]
        },
        {
          "href": "/temp",
          "rt": ["oic.r.temperature"],
          "if": ["oic.if.baseline", "oic.if.s"]
        }
      ],
      "permission": 31,
      "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" ]
        }
      ]
    },
    {
      "aceid": 2,
      "subject": {
        "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
        "role": "SOME_STRING"
      },
```



```
                    "resources": [
                      {
                        "href": "/light",
                        "rt": ["oic.r.light"],
                        "if": ["oic.if.a"]
                      },
                      {
                        "href": "/door",
                        "rt": ["oic.r.door"],
                        "if": ["oic.if.a"]
                      }
                    ],
                    "permission": 15
                  }
                ],
                 "signature": {
                   "sigtype": "oic.sec.sigtype.pk7",
                   "sigvalue": "ENCODED-SIGNATURE-VALUE"
                 }
               }

post:
  description: |
    Sets the sacl resource data

  body:
    application/json:
      schema: |
        {
          "$schema": "http://json-schema.org/draft-04/schema#",
          "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.sacl.json#",
          "title": "Signed Access Control List information",
          "definitions": {
            "oic.r.sacl": {
              "type": "object",
              "properties": {
```



```
                "aclist2":  {
                  "type": "array",
                  "description": "Access Control Entries in the Acl resource",
                  "$ref": "oic.sec.ace2.json#/definitions/oic.sec.ace2"
                },
                "signature":   {
                  "type": "object",
                  "description": "The signature over the ACL resource",
                  "$ref": "oic.sec.sigtype.json#/definitions/oic.sec.sigtype"
                }
              }
            }
          },
          "type": "object",
          "allOf": [
            { "$ref": "#/definitions/oic.r.sacl" }
          ],
          "required": ["aclist2", "signature"],
          "additionalItems": false
        }

      example: /

        {
          "aclist2": [
              {
                "aceid": 1,
                "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
                "resources": [
                  {
                    "href": "/temp",
                    "rt": ["oic.r.temperature"],
                    "if": ["oic.if.baseline", "oic.if.a"]
                  },
                  {
                    "href": "/temp",
                    "rt": ["oic.r.temperature"],
```



```
                    "if": ["oic.if.baseline", "oic.if.s"]
                  }
                ],
                "permission": 31,
                "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" ]
                  }
                ]
              },
              {
                "aceid": 2,
                "subject": {
                    "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
                    "role": "SOME_STRING"
                },
                "resources": [
                  {
                    "href": "/light",
                    "rt": ["oic.r.light"],
                    "if": ["oic.if.baseline", "oic.if.a"]
                  },
                  {
                    "href": "/door",
                    "rt": ["oic.r.door"],
                    "if": ["oic.if.baseline", "oic.if.a"]
                  }
                ],
                "permission": 15
              }
            ],
```



```
4842              "signature": {
4843                "sigtype": "oic.sec.sigtype.pk7",
4844                "sigvalue": "ENCODED-SIGNATURE-VALUE"
4845              }
4846            }
4847
4848      responses :
4849        400:
4850          description: |
4851            The request is invalid.
4852
4853        201:
4854          description: |
4855            The ACL entry is created.
4856
4857        204:
4858          description: |
4859            The ACL entry is updated.
4860
4861    put:
4862      description: |
4863        Sets the sacl resource data
4864
4865      body:
4866        application/json:
4867          schema: |
4868            {
4869              "$schema": "http://json-schema.org/draft-04/schema#",
4870              "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.sacl.json#",
4871              "title": "Signed Access Control List information",
4872              "definitions": {
4873                "oic.r.sacl": {
4874                  "type": "object",
```



```
        "properties": {
          "aclist2":  {
            "type": "array",
            "description": "Access Control Entries in the Acl resource",
            "$ref": "oic.sec.ace2.json#/definitions/oic.sec.ace2"
          },
          "signature":   {
            "type": "object",
            "description": "The signature over the ACL resource",
            "$ref": "oic.sec.sigtype.json#/definitions/oic.sec.sigtype"
          }
        }
      }
    },
    "type": "object",
    "allOf": [
      { "$ref": "#/definitions/oic.r.sacl" }
    ],
    "required": ["aclist2", "signature"],
    "additionalItems": false
  }

example: /

  {
    "aclist2":[
      {
        "aceid": 1,
        "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
        "resources": [
          {
            "href": "/temp",
            "rt": ["oic.r.temperature"],
            "if": ["oic.if.baseline", "oic.if.a"]
          },
          {
            "href": "/temp",
```



```
                        "rt": ["oic.r.temperature"],
                        "if": ["oic.if.baseline", "oic.if.s"]
                      }
                    ],
                    "permission": 31,
                    "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" ]
                      }
                    ]
                  },
                  {
                    "aceid": 2,
                    "subject": {
                        "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
                        "role": "SOME_STRING"
                    },
                    "resources": [
                      {
                        "href": "/light",
                        "rt": ["oic.r.light"],
                        "if": ["oic.if.baseline", "oic.if.a"]
                      },
                      {
                        "href": "/door",
                        "rt": ["oic.r.door"],
                        "if": ["oic.if.baseline", "oic.if.a"]
                      }
                    ],
                    "permission": 15
                  }
```



```
4948                ],
4949                "signature": {
4950                  "sigtype": "oic.sec.sigtype.pk7",
4951                  "sigvalue": "ENCODED-SIGNATURE-VALUE"
4952                }
4953              }
4954
4955        responses :
4956          400:
4957            description: |
4958              The request is invalid.
4959
4960          201:
4961            description: |
4962              The signed ACL entry is created.
4963
4964      delete:
4965        description: |
4966          Deletes the signed ACL data.
4967          When DELETE is used without query parameters, the entire collection is deleted.
4968          When DELETE is used with the query parameter where "acl2" is specified, only the matched entry is
4969    deleted.
4970
4971        queryParameters:
4972          subject:
4973            type: string
4974            description: Delete the signed ACL identified by the string containing subject UUID.
4975
4976            required: false
4977            example: DELETE /mysacl?acl2="de305d54-75b4-431b-adb2-eb6b9e546014, ..."
4978
4979        responses :
4980          200:
```



```
        description: |
          The signed ACL instance or the the entire signed ACL resource has been successfully deleted.

      400:
        description: |
          The request is invalid.
```

A.4.5 Property 정의

Property name	Value type	필수	액세스 모드	설명
aclist2	배열: schema 참조	예		Acl resource 내의 Access Control Entry.
signature	객체: schema 참조	예		ACL resource 에 대한 서명.

A.4.6 CRUDN 동작

Resource	Create	Read	Update	Delete	Notify
/oic/sec/sacl	put	get	post	delete	

A.4.7 참조된 JSON schema

A.4.8 oic.sec.sigtype.json

A.4.9 Property 정의

Property name	Value type	필수	액세스 모드	설명
sigtype	스트링	예		사전 정의된 서명 형식을 지정하는 스트링.
sigvalue	스트링	예		인코딩된 서명.

A.4.10 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.sigtype.json#",
  "title": "Signature format for signed ACL resources",
  "definitions": {
    "oic.sec.sigtype": {
```



```
        "description": "Encoded signature data",
        "properties": {
          "sigtype": {
            "type": "string",
            "enum": ["oic.sec.sigtype.jws", "oic.sec.sigtype.pk7", "oic.sec.sigtype.cws"],
            "description": "The string specifies the predefined signature format",
            "detail-desc": [  "RFC7515 JSON web signature (JWS) object",
                              "RFC2315 base64 encoded object",
                              "CBOR encoded JWS object" ]
          },
          "sigvalue": {
            "type": "string",
            "description": "The encoded signature"
          }
        },
        "required": [ "sigtype", "sigvalue" ]
      }
    }
}
```

A.5 OICSecurityDoxmResource

A.5.1 개요

이 resource 는 device 소유자를 구성하는데 필요한 property 를 규정한다.

A.5.2 URI 예

/oic/sec/doxm

A.5.3 Resource Type

A.5.4 RAML 정의

```
#%RAML 0.8
title: OICSecurityDoxmResource
version: v1.0-20150819
traits:
 - interface :
     queryParameters:
```



```
        if:
          enum: ["oic.if.baseline"]

/oic/sec/doxm:
  description: |
    This resource specifies properties needed to establish a device owner.

  is : ['interface']
  get:
    description: |
      Retrieves the DOXM resource data.

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.doxm.json#",
                "title": "Device Owner Transfer Method information",
                "definitions": {
                  "oic.r.doxm": {
                    "type": "object",
                    "properties": {
                      "oxms":  {
                        "type": "array",
                        "readOnly": true,
                        "description": "List of supported owner transfer methods",
                        "items": {
                          "$ref": "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
                        }
                      },
                      "oxmsel":   {
```



```
                "description": "The selected owner transfer method used during on-boarding",
                "$ref": "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
              },
              "sct": {
                "readOnly": true,
                "description": "Bitmask encoding of supported credential types",
                "$ref": "oic.sec.credtype.json#/definitions/oic.sec.credtype/properties/bitmask"
              },
              "owned":   {
                "type": "boolean",
                "description": "Ownership status flag"
              },
              "deviceuuid": {
                "description": "The uuid formatted identity of the device",
                "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
              },
              "devowneruuid": {
                "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
              },
              "rowneruuid": {
                "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
              }
            }
          }
        },
        "type": "object",
        "allOf": [
          { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
          { "$ref": "#/definitions/oic.r.doxm" }
        ],
        "required": [ "oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid", "rowneruuid" ]
      }

    example: /

      {
        "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"
              }

      400:
        description: |
          The request is invalid.

  post:
    description: |
      Updates the DOXM resource data

    body:
      application/json:
        schema: |
          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.doxm.json#",
            "title": "Device Owner Transfer Method information",
            "definitions": {
              "oic.r.doxm": {
                "type": "object",
                "properties": {
                  "oxms":  {
                    "type": "array",
                    "readOnly": true,
                    "description": "List of supported owner transfer methods",
                    "items": {
                      "$ref": "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
                    }
                  },
```



```
                "oxmsel":   {
                  "description": "The selected owner transfer method used during on-boarding",
                  "$ref": "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
                },
                "sct": {
                  "readOnly": true,
                  "description": "Bitmask encoding of supported credential types",
                  "$ref": "oic.sec.credtype.json#/definitions/oic.sec.credtype/properties/bitmask"
                },
                "owned":   {
                  "type": "boolean",
                  "description": "Ownership status flag"
                },
                "deviceuuid": {
                  "description": "The uuid formatted identity of the device",
                  "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
                },
                "devowneruuid": {
                  "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
                },
                "rowneruuid": {
                  "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
                }
              }
            }
          },
          "type": "object",
          "allOf": [
            { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
            { "$ref": "#/definitions/oic.r.doxm" }
          ],
          "required": [ "oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid", "rowneruuid" ]
        }
```

example: /


```
            {
              "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"
            }

    responses :
      400:
        description: |
          The request is invalid.

      201:
        description: |
          The DOXM entry is created.

      204:
        description: |
          The DOXM entry is updated.

  put:
    description: |
      Creates the DOXM resource data

    body:
      application/json:
        schema: /
          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.doxm.json#",
```



```
"title": "Device Owner Transfer Method information",
"definitions": {
  "oic.r.doxm": {
    "type": "object",
    "properties": {
      "oxms":  {
        "type": "array",
        "readOnly": true,
        "description": "List of supported owner transfer methods",
        "items": {
          "$ref": "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
        }
      },
      "oxmsel":   {
        "description": "The selected owner transfer method used during on-boarding",
        "$ref": "oic.sec.doxmtype.json#/definitions/oic.sec.doxmtype/properties/oxm"
      },
      "sct": {
        "readOnly": true,
        "description": "Bitmask encoding of supported credential types",
        "$ref": "oic.sec.credtype.json#/definitions/oic.sec.credtype/properties/bitmask"
      },
      "owned":   {
        "type": "boolean",
        "description": "Ownership status flag"
      },
      "deviceuuid": {
        "description": "The uuid formatted identity of the device",
        "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
      },
      "devowneruuid": {
        "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
      },
      "rowneruuid": {
        "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
      }
    }
```



```
          }
        },
        "type": "object",
        "allOf": [
          { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
          { "$ref": "#/definitions/oic.r.doxm" }
        ],
        "required": [ "oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid", "rowneruuid" ]
      }

    example: |
      {
        "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"
      }

  responses :
    400:
      description: |
        The request is invalid.

    201:
      description: |
        The DOXM entry is created.

delete:
  description: |
    Deletes the DOXM data.
    When DELETE is used without query parameters, the entire DOXM resource is deleted.
```



```
      When DELETE uses a query value naming the deviceid, only the matched entry is deleted.

    queryParameters:
      subject:
        type: string
        description: Delete the DOXM entry identified by the string containing device ID.

        required: false
        example: DELETE /mydoxm?deviceuuid="de305d54-75b4-431b-adb2-eb6b9e546014"

    responses :
      400:
        description: |
          The request is invalid.

      204:
        description: |
          The CRL instance or the the entire CRL resource has been successfully deleted.
```

A.5.5 Property 정의

Property name	Value type	필수	액세스 모드	설명
rowneruuid	복수 타입: schema 참조	예		
oxms	배열: schema 참조	예	읽기 전용	지원되는 소유권 이전 방법의

				목록.
devowneruuid	복수 타입: schema 참조	예		
deviceuuid	복수 타입: schema 참조	예		device 의 uuid 형식의 아이덴티티.
owned	boolean	예		소유권 상태 플래그.
oxmsel	복수 타입: schema 참조	예		온보딩 시의 사용에 선택된 소유권 이전 방법.
sct	복수 타입: schema 참조	예	읽기 전용	지원되는 크리덴셜 타입의 비트마스크 인코딩.

A.5.6 CRUDN 동작

Resource	Create	Read	Update	Delete	Notify
/oic/sec/doxm	put	get	post	delete	

A.5.7 참조된 JSON schema

A.5.8 oic.sec.doxmtype.json

A.5.9 Property 정의

Property name	Value type	필수	액세스 모드	설명
oxm	정수	예		각각의 값은 특정 소유권 이전 방법을 나타낸다.

A.5.10 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.doxmtype.json#",
  "title": "Device Owner Transfer Method Types",
  "definitions": {
    "oic.sec.doxmtype": {
      "type": "object",
```



```
        "description": "The device owner transfer methods that may be selected at device on-boarding",
        "properties": {
          "oxm": {
            "type": "integer",
            "description": "Each value indicates a specific Owner Transfer method",
            "detail-desc": [  "0 - 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": "object",
    "allOf": [
      { "$ref": "#/definitions/oic.sec.doxmtype" }
    ],
    "required": [ "oxm" ]
  }
```

A.5.11 oic.sec.credtype.json

A.5.12 Property 정의

Property name	Value type	필수	액세스 모드	설명
bitmask	정수	예		비트마스크로서 인코딩된 크리덴셜 타입.

A.5.13 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.credtype.json#",
  "title": "Credential Types",
```



```
  "definitions": {
    "oic.sec.credtype": {
      "type": "object",
      "description": "OIC credential types",
      "properties": {
        "bitmask": {
          "type": "integer",
          "minimum": 0,
          "maximum": 63,
          "description": "Cred type encoded as a bitmask ",
          "detail-desc": [  "0 - Empty credential 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" ]
        }
      }
    },
    "type": "object",
    "allOf": [
      { "$ref": "#/definitions/oic.sec.credtype" }
    ],
    "required": ["bitmask"]
  }
```

A.6 OICSecurityPstatResource

A.6.1 개요

이 resource 는 device 프로비저닝 상태를 규정한다.

A.6.2 URI 예

/oic/sec/pstat

A.6.3 Resource Type

A.6.4 RAML 정의

```
#%RAML 0.8
title: OICSecurityPstatResource
version: v1.0-20150819
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.baseline"]

/oic/sec/pstat:
  description: |
    This resource specifies device provisioning status.

  is : ['interface']
  get:
    description: |
      Retrieves device provisioning status data.

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.pstat.json#",
                "title": "Device Provisioning Status information",
                "definitions": {
                  "oic.r.pstat": {
                    "type": "object",
```



```
      "properties": {
        "dos": {
          "type": "object",
          "description": "Device on-boarding state",
          "$ref": "oic.sec.dostype.json#/definitions/oic.sec.dostype"
        },
        "isop":  {
          "type": "boolean",
          "description": "true indicates device is operational"
        },
        "cm":   {
          "description": "Current device provisioning mode",
          "$ref": "oic.sec.dpmtype.json#/definitions/oic.sec.dpmtype/properties/bitmask"
        },
        "tm": {
          "description": "Target device provisioning mode",
          "$ref": "oic.sec.dpmtype.json#/definitions/oic.sec.dpmtype/properties/bitmask"
        },
        "om": {
          "description": "Current operational mode",
          "$ref": "oic.sec.pomtype.json#/definitions/oic.sec.pomtype/properties/bitmask"
        },
        "sm": {
          "readOnly": true,
          "description": "Supported operational modes",
          "$ref": "oic.sec.pomtype.json#/definitions/oic.sec.pomtype/properties/bitmask"
        },
        "rowneruuid": {
          "description": "The UUID formatted identity of the Resource owner",
          "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
        }
      }
    }
  },
  "type": "object",
  "allOf": [
    { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
```



```
                { "$ref": "#/definitions/oic.r.pstat" }
              ],
              "required": [ "dos", "isop", "cm", "om", "sm", "rowneruuid" ]
            }

          example: |
            {
              "dos": {"s": 3, "p": true},
              "isop": true,
              "cm": 8,
              "tm": 60,
              "om": 2,
              "sm": 7,
              "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
            }

      400:
        description: |
          The request is invalid.

  post:
    description: |
      Sets or updates device provisioning status data.

    body:
      application/json:
        schema: |
          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.pstat.json#",
            "title": "Device Provisioning Status information",
            "definitions": {
              "oic.r.pstat": {
                "type": "object",
```

```
      "properties": {
        "dos": {
          "type": "object",
          "description": "Device on-boarding state",
          "$ref": "oic.sec.dostype.json#/definitions/oic.sec.dostype"
        },
        "isop":  {
          "type": "boolean",
          "description": "true indicates device is operational"
        },
        "cm":   {
          "description": "Current device provisioning mode",
          "$ref": "oic.sec.dpmtype.json#/definitions/oic.sec.dpmtype/properties/bitmask"
        },
        "tm": {
          "description": "Target device provisioning mode",
          "$ref": "oic.sec.dpmtype.json#/definitions/oic.sec.dpmtype/properties/bitmask"
        },
        "om": {
          "description": "Current operational mode",
          "$ref": "oic.sec.pomtype.json#/definitions/oic.sec.pomtype/properties/bitmask"
        },
        "sm": {
          "readOnly": true,
          "description": "Supported operational modes",
          "$ref": "oic.sec.pomtype.json#/definitions/oic.sec.pomtype/properties/bitmask"
        },
        "rowneruuid": {
          "description": "The UUID formatted identity of the Resource owner",
          "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
        }
      }
    }
  },
  "type": "object",
  "allOf": [
    { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
```



```
                { "$ref": "#/definitions/oic.r.pstat" }
              ],
              "required": [ "dos", "isop", "cm", "om", "sm", "rowneruuid" ]
            }

          example: /

            {
              "dos": {"s": 3, "p": true},
              "isop": true,
              "cm": 8,
              "tm": 60,
              "om": 2,
              "sm": 7,
              "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
            }

      responses :

        400:

          description: |
            The request is invalid.

        201:

          description: |
            The PSTAT entry is created.

        204:

          description: |
            The PSTAT entry is updated.

    put:

      description: |
        Sets or updates device provisioning status data.
```



```
body:
  application/json:
    schema: |
      {
        "$schema": "http://json-schema.org/draft-04/schema#",
        "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.pstat.json#",
        "title": "Device Provisioning Status information",
        "definitions": {
          "oic.r.pstat": {
            "type": "object",
            "properties": {
              "dos": {
                "type": "object",
                "description": "Device on-boarding state",
                "$ref": "oic.sec.dostype.json#/definitions/oic.sec.dostype"
              },
              "isop":  {
                "type": "boolean",
                "description": "true indicates device is operational"
              },
              "cm":   {
                "description": "Current device provisioning mode",
                "$ref": "oic.sec.dpmtype.json#/definitions/oic.sec.dpmtype/properties/bitmask"
              },
              "tm": {
                "description": "Target device provisioning mode",
                "$ref": "oic.sec.dpmtype.json#/definitions/oic.sec.dpmtype/properties/bitmask"
              },
              "om": {
                "description": "Current operational mode",
                "$ref": "oic.sec.pomtype.json#/definitions/oic.sec.pomtype/properties/bitmask"
              },
              "sm": {
                "readOnly": true,
                "description": "Supported operational modes",
                "$ref": "oic.sec.pomtype.json#/definitions/oic.sec.pomtype/properties/bitmask"
```



```
                },
                "rowneruuid": {
                  "description": "The UUID formatted identity of the Resource owner",
                  "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
                }
              }
            }
          },
          "type": "object",
          "allOf": [
            { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
            { "$ref": "#/definitions/oic.r.pstat" }
          ],
          "required": [ "dos", "isop", "cm", "om", "sm", "rowneruuid" ]
        }

      example: /
        {
          "dos": {"s": 3, "p": true},
          "isop": true,
          "cm": 8,
          "tm": 60,
          "om": 2,
          "sm": 7,
          "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
        }

    responses :
      400:
        description: |
          The request is invalid.

      201:
        description: |
```



```
        The PSTAT entry is created.

  delete:
    description: |
      Deletes the PSTAT data.
      When DELETE is used without query parameters, the entire collection is deleted.
      When DELETE is used with the query parameter where rowneruuid is specified, only the matched
entry is deleted.

    queryParameters:
      subject:
        type: string
        description: Delete the PSTAT identified by the string containing the UUID.

        required: false
        example: DELETE /mypstat?rowneruuid="de305d54-75b4-431b-adb2-eb6b9e546014"

    responses :
      200:
        description: |
          The PSTAT instance or the the entire resource has been successfully deleted.

      400:
        description: |
          The request is invalid.
```

A.6.5 Property 정의

Property name	Value type	필수	액세스 모드	설명

rowneruuid	복수 타입: schema 참조	예		Resource 소유자의 UUID 형식의 아이덴티티
om	복수 타입: schema 참조	예		현재 동작 모드
cm	복수 타입: schema 참조	예		현재 device 프로비저닝 모드
isop	boolean	예		참 값은 device 가 동작 가능함을 나타낸다.
tm	복수 타입: schema 참조	예		타겟 device 프로비저닝 모드
sm	복수 타입: schema 참조	예	Read Only	지원되는 동작 모드
dos	객체: schema 참조	예		Device 온보딩 상태

A.6.6 CRUDN 동작

Resource	Create	Read	Update	Delete	Notify
/oic/sec/pstat	put	get	post	delete	

A.6.7 참조된 JSON schema

A.6.8 oic.sec.dostype.json

A.6.9 Property 정의

Property name	Value type	필수	액세스 모드	설명
p	boolean	예		device resource 에 필요한 모든 변경이 완료될 때까지 's' 상태가 보류 중이면 'p'는 TRUE 이다.
s	정수	예		현재 또는 보류 중인 동작 상태.

A.6.10 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.dostype.json#",
  "title": "Device On-boarding States",
  "definitions": {
    "oic.sec.dostype": {
      "description": "Device operation state machine",
      "properties" : {
        "s": {
          "type": "integer",
          "minimum": 0,
          "maximum": 4,
          "description": "The current or pending operational state",
          "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" ]
        },
        "p": {
          "type": "boolean",
          "default": true,
          "description": "'p' is TRUE when the 's' state is pending until all necessary changes to device
resources are complete."
        }
      },
      "required": [ "s", "p" ]
    }
  }
}
```

A.6.11 oic.sec.dpmtype.json

A.6.12 Property 정의

Property name	Value type	필수	액세스 모드	설명
bitmask	정수	예		이 값은 8 또는

				16 문자 길이를 갖는다. 8 문자만 갖고 있으면 하위 바이트 값을 나타낸다.

A.6.13 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.dpmtype.json#",
  "title": "Device Provisioning Mode Types",
  "definitions": {
    "oic.sec.dpmtype": {
      "description": "Device provisioning mode maintains a bitmask of the possible provisioning states of a device",
      "properties" : {
        "bitmask": {
          "type": "integer",
          "minimum": 0,
          "maximum": 255,
          "description": "The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value",
          "detail-desc": [  "1 - Manufacturer reset state",
                            "2 - Device pairing and owner transfer state",
                            "4 - Provisiong of bootstrap services",
                            "8 - Provisioning of credential management services",
                            "16 - Provisioning of access management services",
                            "32 - Provisioning of local ACLs",
                            "64 - Initiate Software Version Validation",
                            "128 - Initiate Secure Software Update" ]
        }
      }
    }
  },
  "type": "object",
  "$ref": "#/definitions/oic.sec.dpmtype",
```



```
  "required": [ "bitmask" ]
}
```

A.6.14 oic.sec.pomtype.json

A.6.15 Property 정의

Property name	Value type	필수	액세스 모드	설명
bitmask	정수	예		이 값은 정수로 인코딩된 비트마스크로, 프로비저닝 동작 모드를 나타낸다.

A.6.16 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.pomtype.json#",
  "title": "Provisioning Operational Modes",
  "definitions": {
    "oic.sec.pomtype": {
      "description": "Device provisioning operation may be server directed or client (aka provisioning service) directed.",
      "properties" : {
        "bitmask": {
          "type": "integer",
          "minimum": 1,
          "maximum": 7,
          "description": "The value is a bitmask encoded as integer and indicates the provisioning operation modes",
          "detail-desc": [  "1 - Server-directed utilzing multiple provisioning services",
                            "2 - Server-directed utilzing a single provisioning service",
                            "4 - Client-directed provisioning",
                            "8 - Unused",
                            "16 - Unused",
                            "32 - Unused",
                            "64 - Unused",
                            "128 - Unused" ]
```



```
          }
        }
      }
    },
    "type": "object",
    "allOf": [
      { "$ref": "#/definitions/oic.sec.pomtype" }
    ],
    "required": ["bitmask"]
  }
```

A.6.17

A.7 OICSecurityCredentialResource

A.7.1 개요

이 resource 는 device 가 보안 통신을 구성하는데 사용하는 크리덴셜을 규정한다.

A.7.2 URI 예

/oic/sec/cred

A.7.3 Resource Type

A.7.4 RAML 정의

```
#%RAML 0.8
title: OICSecurityCredentialResource
version: v1.0-20150819
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.baseline"]
 - cred-filtered :
     queryParameters:
       credid:

/oic/sec/cred:
```



```
  description: |
    This resource specifies credentials a device may use to establish secure communication.

  is : ['interface']
  get:
    description: |
      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.

    is : ['cred-filtered']
    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.cred.json#",
                "title": "Device Credentials information",
                "definitions": {
                  "oic.r.cred": {
                    "type": "object",
                    "properties": {
                      "creds": {
                        "type": "array",
                        "description": "List of credentials available at this resource",
                        "items": {
                          "$ref": "oic.sec.cred.json#/definitions/oic.sec.cred"
                        }
                      },
                      "rowneruuid": {
```



```
                "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
              }
            }
          }
        },
        "type": "object",
        "allOf": [
          { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
          { "$ref": "#/definitions/oic.r.cred" }
        ],
        "required": [ "creds", "rowneruuid" ]
      }
```

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.base64",
              "data": "BASE-64-ENCODED-VALUE"
            },
            "privatedata": {
              "encoding": "oic.sec.encoding.base64",
              "data": "BASE-64-ENCODED-VALUE",
              "handle": 4
            },
            "optionaldata": {
              "revstat": false,
              "encoding": "oic.sec.encoding.base64",
```



```
                    "data": "BASE-64-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.base64",
                    "data": "BASE-64-ENCODED-VALUE"
                  },
                  "privatedata": {
                    "encoding": "oic.sec.encoding.base64",
                    "data": "BASE-64-ENCODED-VALUE",
                    "handle": 4
                  },
                  "optionaldata": {
                    "revstat": false,
                    "encoding": "oic.sec.encoding.base64",
                    "data": "BASE-64-ENCODED-VALUE"
                  },
                  "period": "20160101T180000Z/20170102T070000Z",
                  "crms": [ "oic.sec.crm.pk10" ]
                }
              ],
              "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
            }

      400:
        description: |
```



```
        The request is invalid.

  post:
    description: |
      Updates the credential resource with the provided credentials.
      Credentials provided in the update with credid(s) not currently in the
      credential resource are added.
      Credentials provided in the update with credid(s) already in the
      credential resource completely replace the creds in the credential
      resource.
      Credentials provided in the update without credid(s) properties are
      added and assigned unique credid(s) in the credential resource.

    body:
      application/json:
        schema: |
          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.cred.json#",
            "title": "Device Credentials information",
            "definitions": {
              "oic.r.cred": {
                "type": "object",
                "properties": {
                  "creds": {
                    "type": "array",
                    "description": "List of credentials available at this resource",
                    "items": {
                      "$ref": "oic.sec.cred.json#/definitions/oic.sec.cred"
                    }
                  },
                  "rowneruuid": {
                    "$ref": "../../core/schemas/oic.types-schema.json#/definitions/uuid"
                  }
                }
              }
```



```
    },
    "type": "object",
    "allOf": [
      { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
      { "$ref": "#/definitions/oic.r.cred" }
    ],
    "required": [ "creds", "rowneruuid" ]
  }
```

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.base64",
          "data": "BASE-64-ENCODED-VALUE"
        },
        "privatedata": {
          "encoding": "oic.sec.encoding.base64",
          "data": "BASE-64-ENCODED-VALUE",
          "handle": 4
        },
        "optionaldata": {
          "revstat": false,
          "encoding": "oic.sec.encoding.base64",
          "data": "BASE-64-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.base64",
                    "data": "BASE-64-ENCODED-VALUE"
                  },
                  "privatedata": {
                    "encoding": "oic.sec.encoding.base64",
                    "data": "BASE-64-ENCODED-VALUE",
                    "handle": 4
                  },
                  "optionaldata": {
                    "revstat": false,
                    "encoding": "oic.sec.encoding.base64",
                    "data": "BASE-64-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.
        When DELETE is used without query parameters, all the cred entries are deleted.
        When DELETE is used with a query parameter, only the entries matching
        the query parameter are deleted.

      is : ['cred-filtered']
      responses :
        400:
          description: |
            The request is invalid.

        204:
          description: |
            The specific credential(s) or the the entire credential resource has been successfully deleted.
```

A.7.5 Property 정의

Property name	Value type	필수	액세스 모드	설명
rowneruuid	복수 타입: schema 참조	예		
creds	배열: schema 참조	예		resource 에서 사용 가능한 크리덴셜의 목록.

A.7.6 CRUDN 동작

Resource	Create	Read	Update	Delete	Notify
/oic/sec/cred		get	post	delete	

A.7.7 참조된 JSON schema

A.7.8 oic.sec.roletype.json

A.7.9 Property 정의

Property name	Value type	필수	액세스 모드	설명
role	스트링	예		식별되는 role 의 ID.
authority	스트링			식별되는 개체의 기관 요소. NULL <Authority>는 로컬 개체 또는 device 이다.

A.7.10 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.roletype.json#",
  "title": "Security Role Types",
  "definitions": {
    "oic.sec.roletype": {
      "type": "object",
      "description": "Security role specified as an <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or device.",
      "properties": {
        "authority": {
          "type": "string",
          "description": "The Authority component of the entity being identified. A NULL <Authority> refers to the local entity or device."
        },
        "role": {
          "type": "string",
          "description": "The ID of the role being identified."
        }
      },
```



```
        "required": ["role"]
      }
    }
}
```

A.7.11 oic.sec.credtype.json

A.7.12 Property 정의

Property name	Value type	필수	액세스 모드	설명
bitmask	정수	예		비트마스크로서 인코딩된 크리덴셜 타입.

A.7.13 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.credtype.json#",
  "title": "Credential Types",
  "definitions": {
    "oic.sec.credtype": {
      "type": "object",
      "description": "OIC credential types",
      "properties": {
        "bitmask": {
          "type": "integer",
          "minimum": 0,
          "maximum": 63,
          "description": "Cred type encoded as a bitmask ",
          "detail-desc": [  "0 - Empty credential 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" ]
        }
      }
```



```
    }
  },
  "type": "object",
  "allOf": [
    { "$ref": "#/definitions/oic.sec.credtype" }
  ],
  "required": ["bitmask"]
}
```

A.7.14 oic.sec.pubdatatype.json

A.7.15 Property 정의

Property name	Value type	필수	액세스 모드	설명
data	스트링			인코딩된 값.
encoding	스트링			pubdata 에 포함된 데이터의 인코딩 형식을 지정하는 스트링.

A.7.16 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.pubdatatype.json#",
  "title": "Public Credential data",
  "definitions": {
    "oic.sec.pubdatatype": {
      "type": "object",
      "properties": {
        "encoding": {
          "type": "string",
          "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" ],
          "description": "A string specifying the encoding format of the data contained in the pubdata",
          "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" ]
      },
      "data": {
        "type": "string",
        "description": "The encoded value"
      }
    }
  }
}
```

A.7.17 oic.sec.privdatatype.json

A.7.18 Property 정의

Property name	Value type	필수	액세스 모드	설명
handle	정수			키 저장 resource 에 대한 핸들.
data	스트링			인코딩된 값.
encoding	스트링	예		privdata 에 포함된 데이터의 인코딩 형식을 지정하는 스트링.

A.7.19 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.privdatatype.json#",
  "title": "Private Credential data",
  "definitions": {
    "oic.sec.privdatatype": {
      "type": "object",
      "description": "Credential resource non-public contents",
      "properties": {
```



```
        "encoding": {
          "type": "string",
          "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" ],
          "description": "A string specifying the encoding format of the data contained in the privdata",
          "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" ]
        },
        "data": {
          "type": "string",
          "description": "The encoded value"
        },
        "handle": {
          "type": "integer",
          "description": "Handle to a key storage resource"
        }
      },
      "required": [ "encoding" ]
    }
  }
}
```

A.7.20 oic.sec.optdatatype.json

A.7.21 Property 정의

Property name	Value type	필수	액세스 모드	설명
revstat	boolean	예		폐기 상태 플래그 - 참 = 폐기.
data	스트링			인코딩된 구조.
encoding	스트링			optdata 에 포함된 데이터의

				인코딩 형식을 지정하는 스트링.

A.7.22 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.optdatatype.json#",
  "title": "Optional Credential data",
  "definitions": {
    "oic.sec.optdatatype": {
      "description": "Optional credential contents describes revocation status for this credential",
      "properties": {
        "revstat": {
          "type": "boolean",
          "description": "Revocation status flag - true = revoked"
        },
        "encoding": {
          "type": "string",
          "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" ],
          "description": "A string specifying the encoding format of the data contained in the optdata",
          "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" ]
        },
        "data": {
          "type": "string",
          "description": "The encoded structure"
        }
      },
      "required": [ "revstat" ]
    }
```



```
  }
}
```

A.7.23 oic.sec.crmtype.json

A.7.24 Property 정의

Property name	Value type	필수	액세스 모드	설명
crm	스트링	예		각 열거는 크리덴셜을 갱신하는 방법을 나타낸다.

A.7.25 Schema 정의

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.sec.crmtype.json#",
  "title": "Credential Refresh Methods",
  "definitions": {
    "oic.sec.crmtype": {
      "properties": {
        "crm": {
          "type": "string",
          "enum": [ "oic.sec.crm.pro", "oic.sec.crm.psk", "oic.sec.crm.rdp", "oic.sec.crm.skdc",
"oic.sec.crm.pk10" ],
          "description": "Each enum represents a method by which the credentials are
refreshed.",
          "detail-desc": [  "oic.sec.crm.pro - Credentials refreshed by a provisioning service",
                            "oic.sec.crm.rdp - Credentials refreshed by a key agreement protocol
and random PIN",
                            "oic.sec.crm.psk - Credentials refreshed by a key agreement protocol",
                            "oic.sec.crm.skdc - Credentials refreshed by a key distribution service",
                            "oic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a
CA"
          ]
        }
      }
    }
```



```
  },
  "type": "object",
  "allOf": [
    { "$ref": "#/definitions/oic.sec.crmtype" }
  ],
  "required": ["crm"]
}
```

A.8 OICSecurityCsrResource

A.8.1 개요

이 resource 는 Certificate Signing Request 를 규정한다.

A.8.2 URI 예

/oic/sec/csr

A.8.3 Resource Type

A.8.4 RAML 정의

```
#%RAML 0.8
title: OICSecurityCsrResource
version: v1.0-20150819
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.baseline"]

/oic/sec/csr:
  description: |
    This resource specifies a Certificate Signing Request.

  is : ['interface']
  get:
    description: |
      Retrieves CSR data.

    responses :
      200:
        body:
          application/json:
            schema: |
```



```
                {
                  "$schema": "http://json-schema.org/draft-04/schema#",
                  "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.csr.json#",
                  "title": "Certificate Signing Request information",
                  "definitions": {
                    "oic.r.csr": {
                      "type": "object",
                      "properties": {
                        "csr":  {
                          "type": "string",
                          "description": "Signed CSR in ASN.1 in the encoding specified by the
encoding property"
                        },
                        "encoding": {
                          "type": "string",
                          "enum": [ "oic.sec.encoding.pem", "oic.sec.encoding.der" ],
                          "description": "A string specifying the encoding format of the data
contained in csr",
                          "detail-desc": [  "oic.sec.encoding.pem - Encoding for PEM encoded CSR",
                                            "oic.sec.encoding.der - Encoding for DER encoded CSR" ]
                        }
                      }
                    }
                  },
                  "type": "object",
                  "allOf": [
                    { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
                    { "$ref": "#/definitions/oic.r.csr" }
                  ],
                  "required": ["csr","encoding"]
                }

              example: |
                {
                "rt": ["oic.r.csr"],
                "encoding" : "oic.sec.encoding.pem",
                "csr": "PEMENCODEDCSR"
                }

        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.
```

A.8.5 Property 정의

Property name	Value type	필수	액세스 모드	설명
encoding	스트링	예		csr 에 포함된 데이터의 인코딩 형식을 지정하는 스트링.

csr	스트링	예		인코딩 property 에 의해 지정된 인코딩에서 ASN.1 내의 서명된 CSR.

A.8.6 CRUDN 동작

Resource	Create	Read	Update	Delete	Notify
/oic/sec/csr		get			

A.9 OICSecurityRolesResource

A.9.1 개요

이 resource 는 주장된 role 을 규정한다.

A.9.2 URI 예

/oic/sec/roles

A.9.3 Resource Type

A.9.4 RAML 정의

```
#%RAML 0.8
title: OICSecurityRolesResource
version: v1.0-20170323
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.baseline"]

/oic/sec/roles:
  description: |
    This resource specifies roles that have been asserted.

  is : ['interface']
  get:
    description: |
      Retrieves the asserted roles data.

    responses :
      200:
        body:
          application/json:
            schema: |
```



```
                {
                  "$schema": "http://json-schema.org/draft-04/schema#",
                  "id": "https://www.openconnectivity.org/ocf-
apis/security/schemas/oic.r.roles.json#",
                  "title": "Asserted Role Certificates",
                  "definitions": {
                    "oic.r.cred": {
                      "type": "object",
                      "properties": {
                        "roles": {
                          "type": "array",
                          "description": "List of role certificates",
                          "items": {
                            "$ref": "oic.sec.cred.json#/definitions/oic.sec.cred"
                          }
                        }
                      }
                    }
                  },
                  "type": "object",
                  "allOf": [
                    { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
                    { "$ref": "#/definitions/oic.r.cred" }
                  ],
                  "required": [ "roles" ]
                }

              example: |
                {
                  "roles" :[ #array of oic.sec.cred objects
                      {
                        "credid":1,
                        "credtype":8,  # Cred type 8 is SIGNED_ASYMMETRIC_KEY
                        "subjectuuid":"00000000-0000-0000-0000-000000000000",
                        "pbData":       # Role cert, type oic.sec.pubdatatype
                          {
                             "encoding":"oic.sec.encoding.pem",
                             "data":"PEMENCODEDROLECERT"
                          },
                        "optData":        # Optional issuer certificate, type oic.sec.pubdatatype
                          {
                             "encoding":"oic.sec.encoding.pem",
                             "data":"PEMENCODEDISSUERCERT"
                          }
                      },
                      {
                        "credid":2,
                        "credtype":8,  # Cred type 8 is SIGNED_ASYMMETRIC_KEY
                        "subjectuuid":"00000000-0000-0000-0000-000000000000",
                        "pbData":         # Role cert, type oic.sec.pubdatatype
                          {
                             "encoding":"oic.sec.encoding.pem",
                             "data":"PEMENCODEDROLECERT"
                          },
                        "optData":        # Optional issuer certificate, type oic.sec.pubdatatype
                          {
                             "encoding":"oic.sec.encoding.pem",
                             "data":"PEMENCODEDISSUERCERT"
                          }
                      }
                  ],
                  "rt":["oic.sec.cred"],
```



```
                "if":["oic.if.baseline"]
                 }
               }

    400:
      description: |
        The request is invalid.

post:
  description: |
    Update the roles resource, i.e., assert new roles to this server.
    New role certificates that match an existing certificate (i.e., pbData
    and optData are the same) are not added to the resource (and 204 is
    returned).
    The provided credid values are ignored, the resource assigns its own.

  body:
    application/json:
      schema: |
        {
          "$schema": "http://json-schema.org/draft-04/schema#",
          "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.roles.json#",
          "title": "Asserted Role Certificates",
          "definitions": {
            "oic.r.cred": {
              "type": "object",
              "properties": {
                "roles": {
                  "type": "array",
                  "description": "List of role certificates",
                  "items": {
                    "$ref": "oic.sec.cred.json#/definitions/oic.sec.cred"
                  }
                }
              }
            }
          },
          "type": "object",
          "allOf": [
            { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
            { "$ref": "#/definitions/oic.r.cred" }
          ],
          "required": [ "roles" ]
        }

      example: |
        {
          "roles" :[ #array of oic.sec.cred objects
              {
                "credid":1,    # Optional
                "credtype":8,  # Cred type 8 is SIGNED_ASYMMETRIC_KEY
                "subjectuuid":"00000000-0000-0000-0000-000000000000",
                "pbData":       # Role cert, type oic.sec.pubdatatype
                  {
                     "encoding":"oic.sec.encoding.pem",
                     "data":"PEMENCODEDROLECERT"
                  },
                "optData":        # Optional issuer certificate, type oic.sec.pubdatatype
                  {
```



```
                        "encoding":"oic.sec.encoding.pem",
                        "data":"PEMENCODEDISSUERCERT"
                      }
                  },
                  {
                    "credid":2,    #Optional
                    "credtype":8,  # Cred type 8 is SIGNED_ASYMMETRIC_KEY
                    "subjectuuid":"00000000-0000-0000-0000-000000000000",
                    "pbData":         # Role cert, type oic.sec.pubdatatype
                      {
                        "encoding":"oic.sec.encoding.pem",
                        "data":"PEMENCODEDROLECERT"
                      },
                    "optData":        # Optional issuer certificate, type oic.sec.pubdatatype
                      {
                        "encoding":"oic.sec.encoding.pem",
                        "data":"PEMENCODEDISSUERCERT"
                      }
                  }
              ],
              "rt":["oic.sec.cred"],
              "if":["oic.if.baseline"]
              }
```

```
    responses :
      400:
        description: |
          The request is invalid.

      204:
        description: |
          The roles entry is updated.

  delete:
    description: |
      Deletes roles resource entries.
      DELETE does not support query parameters, all the entries are deleted.

    responses :
      200:
        description: |
          The roles resource has been successfully deleted.

      400:
        description: |
          The request is invalid.
```

A.9.5 Property 정의

Property name	Value type	필수	액세스 모드	설명
rowneruuid	복수 타입: schema 참조	에		

creds	배열: schema 참조	예		resource 에서 사용 가능한 크리덴셜의 목록.

A.9.6 CRUDN 동작

Resource	Create	Read	Update	Delete	Notify
/oic/sec/roles		get	post	delete	

A.10 OICSecurityCrlResource

A.10.1 개요

이 resource 는 X.509 객체로서의 인증서 폐기 목록을 규정한다.

A.10.2 URI 예

/oic/sec/crl

A.10.3 Resource Type

A.10.4 RAML 정의

```
#%RAML 0.8
title: OICSecurityCrlResource
version: v1.0-20150819
traits:
 - interface :
     queryParameters:
       if:
         enum: ["oic.if.baseline"]

/oic/sec/crl:
  description: |
    This resource specifies certificate revocation lists as X.509 objects.

  is : ['interface']
  get:
    description: |
      Retrieves crl data.

    responses :
      200:
        body:
          application/json:
            schema: |
              {
                "$schema": "http://json-schema.org/draft-04/schema#",
                "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.crl.json#",
```



```
                "title": "Certificate Revocation List information",
                "definitions": {
                  "oic.r.crl": {
                    "type": "object",
                    "properties": {
                      "crlid":  {
                        "type": "integer",
                        "description": "Local reference to a crl resource"
                      },
                      "thisupdate":   {
                        "type": "string",
                        "description": "UTC time of last CRL update"
                      },
                      "crldata": {
                        "type": "string",
                        "description": "Base64 BER encoded crl data"
                      }
                    }
                  }
                },
                "type": "object",
                "allOf": [
                  { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
                  { "$ref": "#/definitions/oic.r.crl" }
                ],
                "required": ["crlid","thisupdate", "crldata"]
              }

            example: |

              {
              "rt": ["oic.r.crl"],
              "crlid": 1,
              "thisupdate": "2016-04-12T23:20:50.52Z",
              "crldata": "Base64ENCODEDCRL"
              }

  post:

    description: |
      Updates the CRL data

    body:
      application/json:

        schema: |

          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.crl.json#",
            "title": "Certificate Revocation List information",
            "definitions": {
              "oic.r.crl": {
                "type": "object",
                "properties": {
                  "crlid":  {
                    "type": "integer",
                    "description": "Local reference to a crl resource"
                  },
                  "thisupdate":   {
                    "type": "string",
                    "description": "UTC time of last CRL update"
                  },
                  "crldata": {
```



```
                    "type": "string",
                    "description": "Base64 BER encoded crl data"
                  }
                }
              }
            },
            "type": "object",
            "allOf": [
              { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
              { "$ref": "#/definitions/oic.r.crl" }
            ],
            "required": ["crlid","thisupdate", "crldata"]
          }

        example: |
          {
          "rt": ["oic.r.crl"],
          "crlid": 1,
          "thisupdate": "2016-04-12T23:20:50.52Z",
          "crldata": "Base64ENCODEDCRL"
          }

    responses :
      400:
        description: |
          The request is invalid.

      201:
        description: |
          The CRL entry is created.

      204:
        description: |
          The CRL entry is updated.

  put:
    description: |
      Creates the CRL data

    body:
      application/json:
        schema: |
          {
            "$schema": "http://json-schema.org/draft-04/schema#",
            "id": "https://www.openconnectivity.org/ocf-apis/security/schemas/oic.r.crl.json#",
            "title": "Certificate Revocation List information",
            "definitions": {
              "oic.r.crl": {
                "type": "object",
                "properties": {
                  "crlid":  {
                    "type": "integer",
                    "description": "Local reference to a crl resource"
                  },
                  "thisupdate":   {
                    "type": "string",
```

```
                    "description": "UTC time of last CRL update"
                  },
                  "crldata": {
                    "type": "string",
                    "description": "Base64 BER encoded crl data"
                  }
                }
              }
            },
            "type": "object",
            "allOf": [
              { "$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core" },
              { "$ref": "#/definitions/oic.r.crl" }
            ],
            "required": ["crlid","thisupdate", "crldata"]
          }

        example: |
          {
          "rt": ["oic.r.crl"],
          "crlid": 1,
          "thisupdate": "2016-04-12T23:20:50.52Z",
          "crldata": "Base64ENCODEDCRL"
          }

    responses :
      400:
        description: |
          The request is invalid.

      201:
        description: |
          The CRL entry is created.

  delete:
    description: |
      Deletes the CRL data.
      When DELETE is used without query parameters, the entire CRL resource is deleted.
      When DELETE uses a query value naming the crlid, only the matched entry is deleted.

    queryParameters:
      subject:
        type: string
        description: Delete the CRL identified by the string containing CRL ID.

        required: false
        example: DELETE /mycrl?crlid=0

    responses :
      400:
        description: |
          The request is invalid.

      204:
```



```
        description: |
          The CRL instance or the the entire CRL resource has been successfully deleted.
```

A.10.5 Property 정의

Property name	Value type	필수	액세스 모드	설명
crldata	스트링	예		Base64 BER 인코딩된 crl 데이터.
crlid	정수	예		crl resource 에의 로컬 참조.
thisupdate	스트링	예		마지막 CRL 갱신의 UTC 시각.

A.10.6 CRUDN 동작

Resource	Create	Read	Update	Delete	Notify
/oic/sec/crl	put	get	post	delete	