OCF 2.3 – Device Identity and Certificate Clarification related to section 7.1.1. – Sec WG CR 2691

7.1.1 Device Identity for Devices with UAID [DEPRECATED]

This section intentionally left blank.

~~When a manufacturer certificate is used with certificates chaining to an OCF root CA (as specified in Section 7.1.1), the manufacturer shall include a Platform ID inside the certificate subject CN field. In such cases, the device ID may be created according to the Unique Authenticable IDentifier (UAID) scheme defined in this section.~~

~~For identifying and protecting Devices, the Platform Secure Execution Environment (SEE) may opt to generate new Dynamic Public Key Pair (DPKP) for each Device it is hosting, or it may opt to simply use the same public key credentials embedded by manufacturer; Embedded Platform Credential (EPC). In either case, the Platform SEE will use its Random Number Generator (RNG) to create a device identity called UAID for each Device. The UAID is generated using eitherEPC only or the combnation of DPC and EPC if both are available. When both are available, the Platform shall use both key pairs to generate the UAID as described in this section.~~

~~The Device ID is formed from the device's public keys and associated OCF Cipher Suite. The Device ID is formed by:~~

~~1) Determining the OCF Cipher Suite of the Dynamic Public Key. The Cipher Suite curve must match the usage of the AlgorithmIdentifier used in SubjectPublicKeyInfo as intended for use with Device security mechanisms. Use the encoding of the CipherSuite as the 'csid' value in the following calculations. Note that if the OCF Cipher Suite for Dynamic Public key is different from the ciphersuite indicated in the Platform certificate (EPC), the OCF Cipher Suite shall be used below.~~

~~2) From EPC extract the value of embedded public key. The value should correspond to the value of subjectPublicKey defined in SubjectPublicKeyInfo of the certificate. In the following we refer to this as EPK. If the public key is extracted from a certificate, validate that the AlgorithmIdentifier matches the expected value for the CipherSuite within the certificate.~~

~~3) From DPC Extract the value of the public key. The value should correspond to the value of subjectPublicKey defined in SubjectPublicKeyInfo. In the following we refer to this as DPK.~~

~~4) Using the hash for the Cipher Suite calculate: h = hash('uaid' | csid | EPK| DPK | <other_info>)~~

~~Other_info could be 1) device type as indicated in /oic/d (could be read-only and set by manufacturer), 2) in case there are two sets of public key pairs (one embedded, and one dynamically generated), both public keys would be included.~~

~~5) Truncate to 160 bits by taking the leftmost 160 bits of h UAID = h[0:16] # leftmost 16 octets~~

6) Convert the binary UAID to a ASCII string by 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) Append the string value of USID to 'urn:usid:' to form the final string value of the Device ID urn:usid:ABXW....

Whenever the public key is encoded the format described in RFC 7250 for SubjectPublicKeyInfo shall be used.

Validation of UAID

To be able to use the newly generated Device ID (UAID) and public key pair (DPC), the device Platform shall use the embedded private key (corresponding to manufacturer embedded public key and certificate) to sign a token vouching for the fact that it (the Platform) has in fact generated the DPC and UAID and thus deferring the liability of the use of the DPC to the new device owner. This also allows the ecosystem to extend the trust from manufacturer certificate to a device issued certificate for use in the new DPC and UAID. The degree of trust is in dependent of the level of hardening of the device SEE.

```
Dev_Token=Info, Signature(hash(info))
Signature algorithm=ECDSA (can be same algorithm as that in EPC or that possible for DPC)
Hash algorithm=SHA256
Info=UAID| <Platform ID> | UAID_generation_data | validity
UAID_generation_data=data passed to the hash algorithm used to generate UAID.
Validity=validity period in days (how long the token will be valid)
```

REMOVE reference to this section from the B lack Security Profile:

14.7.2.3.2 Requirements for Certification at Security Profile Black (Normative)

Every device with "currentprofile" Property of the /oic/sec/sp Resource designating a Security Profile of "oic.sec.sp.black", as defined in section 14.7.1, must support each of the following:

- **Onboarding via OCF Rooted Certificate Chain, including PKI chain validation**
- **Support for AES 128 encryption for data at rest and in transit.**
- **Hardening minimums: manufacturer assertion of secure credential storage**
- ~~**In section 7.1.1 in enumerated item #2: "The value should correspond to the value of subjectPublicKey defined in SubjectPublicKeyInfo of the certificate" is changed to require this format: "The value shall correspond to the value of subjectPublicKey defined in SubjectPublicKeyInfo of the certificate. "**~~
- ~~**In section 7.1.1 in the enumerated item #3: "The value should correspond to the value of subjectPublicKey defined in SubjectPublicKeyInfo" is changed to require this format: "The value SHALL correspond to the value of subjectPublicKey defined in SubjectPublicKeyInfo."**~~
- **In section 8.2 in enumerated item #10 "The /oic/sec/cred Resource should contain credential(s) if required by the selected OTM" is changed to require the credential be stored: "The /oic/sec/cred Resource shall contain credential(s)."**