|  |
| --- |
|  |

|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* | SDS 41 |
| Source:\* | Leila Le Brun, Orange ([leila.lebrun@orange.com](mailto:leila.lebrun@orange.com));  Chrystel Gaber, Orange ([chrystel.gaber@orange.com](mailto:chrystel.gaber@orange.com) ) |
| Date:\* | 2019-06-30 |
| Reason for Change/s:\* | To enhace dynamic authorization by adding nested token |
| CR against: Release\* | Rel-4 |
| CR against: WI\* | Active <Work Item number>  MNT maintenance / < Work Item number(optional)>  Is this a mirror CR? Yes  No  mirror CR number: (Note to Rapporteur - use latest agreed revision)  STE Small Technical Enhancements / < Work Item number (optional)>  Only ONE of the above shall be ticked |
| CR against: TS/TR\* | TS-0003 |
| Clauses \* | TS-0003 clause 7.3.2.4, 7.3.2.5, 7.3.2.6.3 |
| Type of change: \* | Editorial change  Bug Fix or Correction  Change to existing feature or functionality  New feature or functionality  Only ONE of the above shall be ticked |
| Other TS/TR(s) impacted | <TS/TR number>, <Version Number>, and <Description on which aspect should be reflected in this TS/TR> |
| Post Freeze checking:\* | This CR contains only essential changes and corrections? YES  NO  This CR may break backwards compatibility with the last approved version of the TS? YES  NO |
| Template Version: January 2019 (do not modify) | |

**oneM2M Notice**

The document to which this cover statement is attached is submitted to oneM2M. Participation in, or attendance at, any activity of oneM2M, constitutes acceptance of and agreement to be bound by terms of the Working Procedures and the Partnership Agreement, including the Intellectual Property Rights (IPR) Principles Governing oneM2M Work found in Annex 1 of the Partnership Agreement.

GUIDELINES for Change Requests:

Provide an informative introduction containing the problem(s) being solved, and a summary list of proposals.

Each CR should contain changes related to only one particular issue/problem.

If this is a correction, and the change applies to previous releases, a separate “mirror CR” should be posted at the same time as this CR

Mirror CR: applies only when the text, including clause numbering are exactly the same.

Companion CR: applies when the change means the same but the baselines differ in some way (e.g. clause number).

Follow the principle of completeness, where all changes related to the issue or problem within a deliverable are simultaneously proposed to be made e.g. a change impacting 5 tables should not only include a proposal to change only 3 tables. Include any changes to references, definitions, and abbreviations in the same deliverable.

Follow the drafting rules.

All pictures must be editable.

Check spelling and grammar.

Use change bars for modifications.

The change should include the current and surrounding clauses to clearly show where a change is located and to provide technical context of the proposed change. Additions of complete clauses need not show surrounding clauses as long as the proposed clause number clearly shows where the proposed new clause is located.

Multiple changes in a single CR shall be clearly separated by horizontal lines with embedded text such as, start of change 1, end of change 1, start of new clause, end of new clause.

When subsequent changes are made to the content of a CR, then the accepted version should not show changes over changes. The accepted version of the CR should only show changes relative to the baseline approved text.

## Introduction

Ofter AEs are connected to several networks and AEs resources are requested and used by different originator AEs/CSEs. Unfrounitely it exposes AEs to the attacks when the hacker using one acces network/CSE to get a full control on AEs and to propage the attack to other networks to which AEs is connected.

Example of this attack if the attack to ‘Target’ retailer when 40 million individuals were exposed and when hackers have sucesseded to get credit card data including account numbers, CVV security codes, and expiration dates

Hackers began with stolen credentials from an HVAC company that acts as a contractor to several Target locations. (HVAC company IT is connected to Air Conditionner’s installed in the Target shops in order to monitor energy consumption and temperatures in the stores to save on costs and to alert store managers if rempertatures in the stored fluctuate otside of an acceptable range that could prevent customers from shopping in the store).

Once entered in the HVAC company IT, hackers have sucessed deployment of the malware on the Air Conditionners which where connected to the internal Target network. Via this internal network, the intruders pushed the card-stealing malicious software to the cash registers within Target stores.

Example Target attack using HVAC

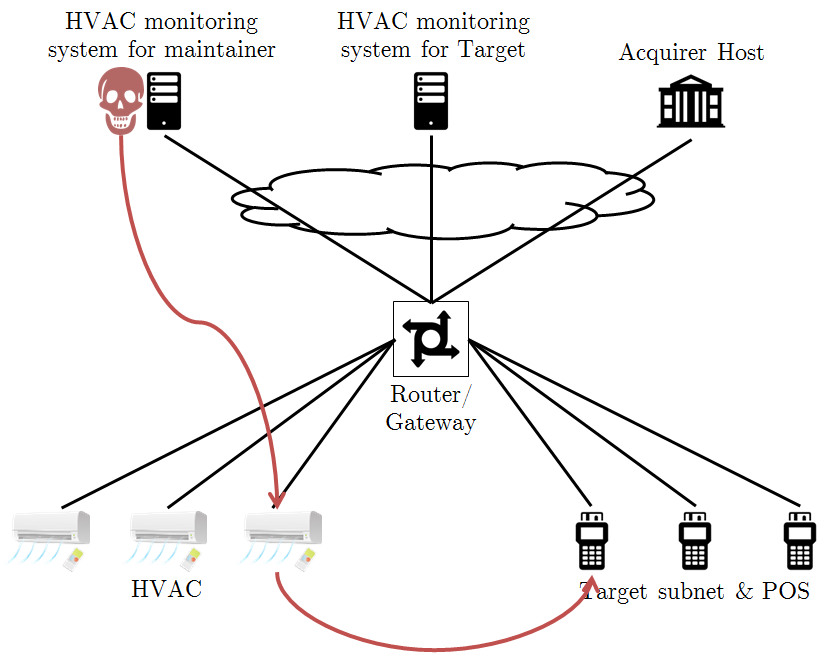


Figure 1: Attack on Target retailer via HVAC air conditionners

To prevent this kind of attacks, ODSI europenian project proposes to add an ASN node and use multi-tenant isolation within it. This will allow giving external company an access to the resources and in the same keep a strict control of ressources by the owner of AEs.

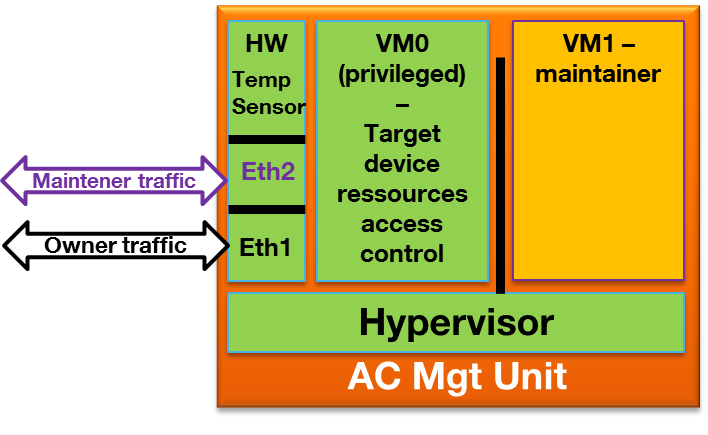


Figure 2: Multi-tenants isolation within the device

In case of multi-tenant isolation, it will be possible to control an access to the sensitive resources of the privileged VM. In order to allow this control, the proposal is to introduce nestedToken which will allow an access to the VM0 sensitive resources Then:

* token will be used to control VM1 resources in case maintener needs to access VM1
* nestedToken will be used to control VM0 resources in case owner need to access VM0

**Multi-Tenant Use Cases**

Air Conditioners are installed as part of Smart Office.

Maintenance of Air Conditioners is subcontracted to a **maintainer (M)** who manages the device resources remotely (humidity level, temperature, on/off, power consumption, firmware version,…).

**Owner (O)** is also able to manage the device remotely (humidity level, temperature,…)

Each actor is associated to a Domain or VM in the device. The hypervisor & owner privileged VM manage the “real” resources and expose “virtual resources” to tenants (M).

Each Domain is associated to a ASN-CSE (ASN-CSE\_O, ASN-CSE\_M).

M has an algorithm for maintenance deployed on the HVAC. M therefore has access to device resources & domain resources (version, results of processing, alerts)

**Owner, Maintainer** each have their own DAS (DAS\_O & DAS\_M). ASN-CSE\_O validates tokens emitted by DAS\_O and ASN-CSE\_M validates tokens emitted by DAS\_M.

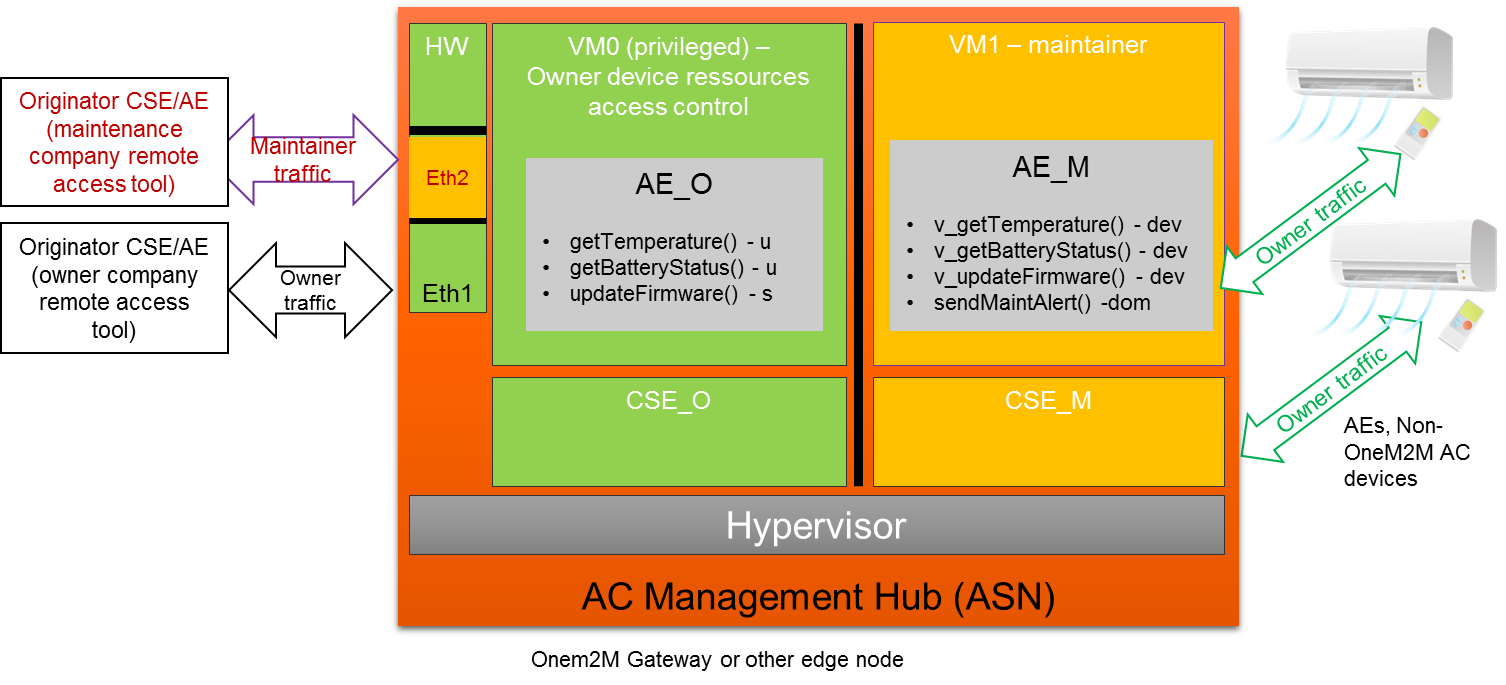


Figure 3: Air Conditionners multi-tenant Use Case

Let’s describe several Use Cases showing that different types of usage will require different tokens to isolate sensitive resources and to give different priviledges.

1. **Case 1:** If the administrator from O (Owner) wants to modify temperature (device resource), then O shall collect a token from DAS\_O. In this case no need for nestedToken:



Figure 4: Case 1, no need for nestedToken

1. **Case 2**: If the administrator from M wants to update some AI parameters (M domain ressource), then M shall collect a token from DAS\_M. In this case no need for nestedToken:

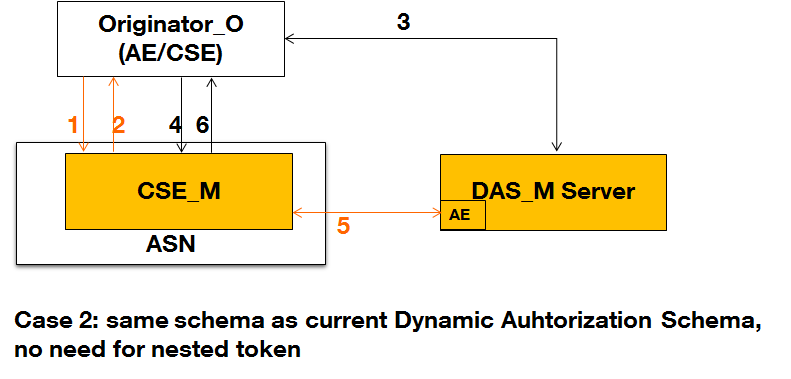


Figure 4: Case 2, no need for nestedToken

1. **Case 3:** If the administrator from M wants to modify the temperature (unsensitive device resource) then M shall collect a token from DAS\_M. In this case no need for nestedToken:

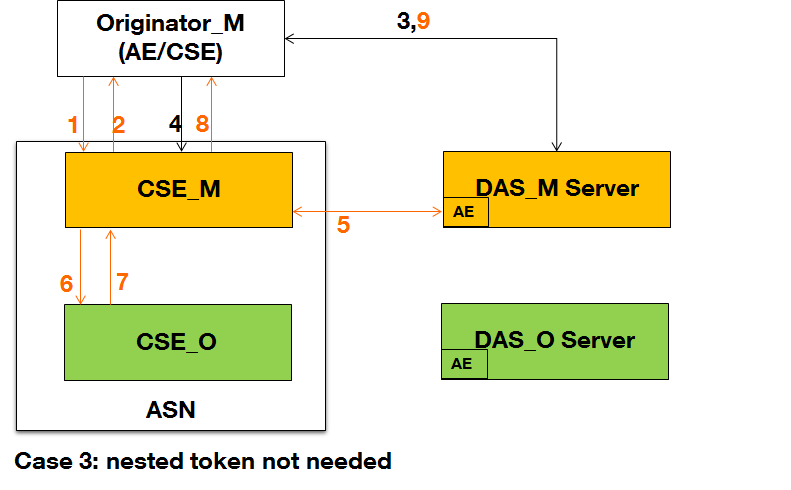


Figure 5: Case 3, no need for nestedToken

1. **Case 4:** If the administrator from M wants to modify the firmware (sensitive device resource), then M shall collect a token from DAS\_M which will collect a token from DAS\_O and include it as a nested token. In this case nestedToken is required:

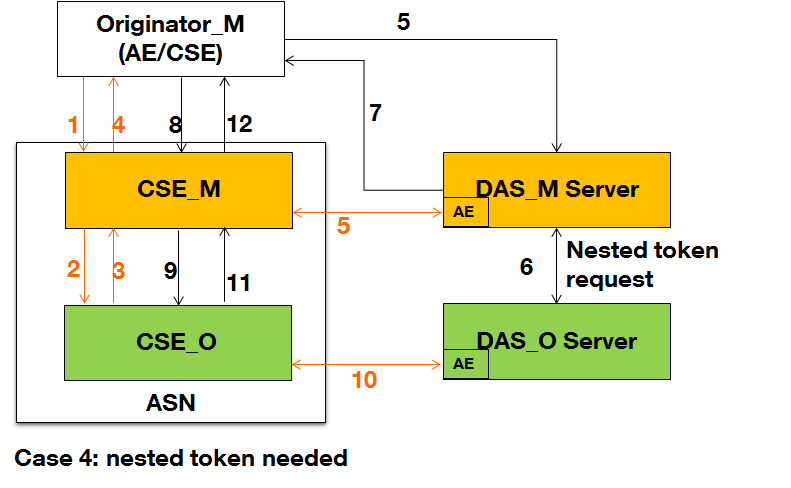


Figure 6: Case 4, nestedToken needed

For each request from M related to a device resource :

- DAS\_M needs to know whether a DAS\_O token is needed or not. DAS\_O can communicate this information through a security policy, documentation or a specific API.

- CSE\_M sends to the hypervisor & CSE\_O a request to read/modify

**Use Cases Summary:**

| **Case N** | **Actor performing an action on AE** | **Case Description** | **DAS providing token** | **nestedToken needed** |
| --- | --- | --- | --- | --- |
| Case 1 | O (Owner) | The administrator from O wants to modify temperature (device resource) | O collects a token from DAS\_O | no |
| Case 2 | M (Maintainer) | The administrator from M wants to update some AI parameters (M domain ressource), | M collects a token from DAS\_M | no |
| Case 3 | M (Maintainer) | The administrator from M wants to modify the temperature (unsensitive device resource) | M collects a token from DAS\_M | no |
| Case 4 | M (maintainer) | The administrator from M wants to modify the firmware (sensitive device resource), | M collects a token from DAS\_M which will collect a token from DAS\_O and include it as a nested token. | yes |

Here are the call flow detailed on the Case 4, when nested Token is required:

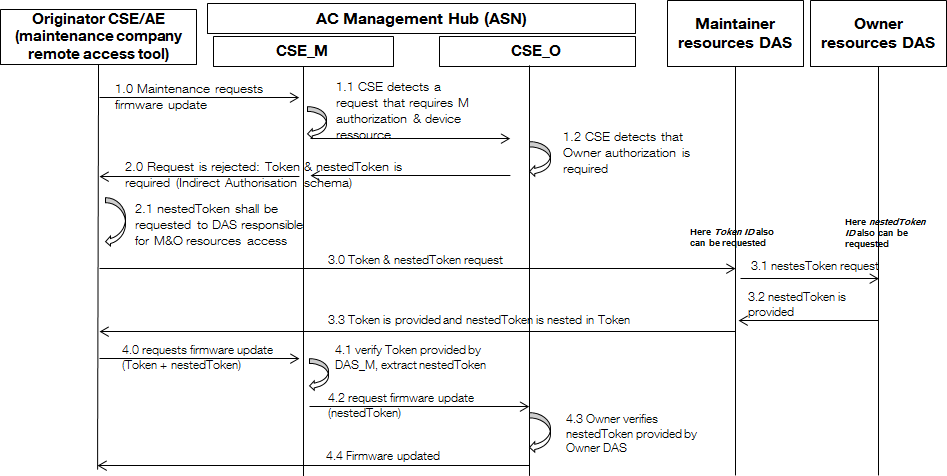


Figure 7: Case 4 Call Flow

-----------------------Start of TS-0003 change 1-------------------------------------------

#### 7.3.2.4 Token Structure

A token is used to carry authorization information that can be roles assigned to the token holder or access control policies applicable to the token holder. The structure of token is shown in figure 7.3.2.4-1, it contains the following data fields:

* version: version of the token.
* tokenID: unique ID of the token.
* holder: ID of the token holder.
* issuer: ID of the token issuer.
* notBefore: token valid from this time.
* notAfter: token expired after this time.
* tokenName: optional, human readable name of the token.
* audience: optional, list of CSE\_IDs of the CSEs expected to accept the token.
* permissions: permissions associated with the token. Its format is specified in clause 9.6.39 of oneM2M TS‑0001 [1].
* extension: used for store other information, e.g. application-specific information.
* nestedToken: in the case of multi-tenancy & virtualized architecture, a token can contain a nested token. The tenant’s application uses resources provided by the Owner of the virtualized equipment. The token proves that the tenant authorization server has authorized the action. The nested token proves that the owner authorization servre has authorized the access to the equipment’s virtualized resource

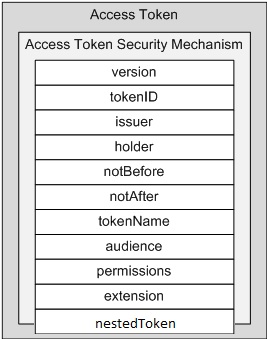


Figure 7.3.2.4-1: Structure of token

A token shall be protected by the ESData security mechanism. A token shall be signed, encrypted or signed and encrypted.

#### Token Evaluation

The generic process of evaluating a token can be described as follows:

1. Token security validation: Depending on the security mechanism used by a token, the validation may be:

* Verifying signed token;
* Decrypting encrypted token; or
* Decrypting and verifying signed and encrypted token.

After passing the token security validation, the plain text of the token can be used for further validation.

1. Token content validation: Depending on the content contained in the token, the validation may check:

* If the identity of the Originator equal to the token holder specified in the *holder* data field.
* If the token issuer specified in the *issuer* data field is valid.
* If this token is not expired according to the *notBefore* and *notAfter* data fields.
* If the identifier of the Hosting CSE is in the CSE-ID list specified by the *audience* data field (in case the *audience* data field is not empty)

After passing the token content validation, the permissions associated to this token shall be used for access control.

1. Token permissions evaluation: Checking the permission element in the permission list one by one until the access request is permitted by one of the permissions or end of the list. For each permission in the list of the permissions the evaluation shall be done as follows:
2. Checking *resourceIDs* element. If it is present, then the authorization information described in *privileges* and/or *roleIDs* elements shall apply only to the resources specified by this element. If the *privileges* element is present, then this element shall be present.
3. If the *privileges* element is present, the access control rules held in this element shall be used as applicable access control policy in the current access control decision making process.
4. If the *roleIDs* element is present, the Role-IDs held in this element shall be used as valid roles in the current access control decision making process.
5. Nested token evaluation: In the case of multi-tenancy & virtualized architecture, a token can contain a nested token. The tenant’s application uses resources provided by the Owner of the virtualized equipment. The token proves that the tenant authorization server has authorized the action. The nested token proves that the owner authorization servre has authorized the access to the equipment’s virtualized resource.
6. Checking *nestedToken* element. If it is present, then its content is retrieved and sent to the Owner’s Virtual Machine along with the request for the virtualized resource.
7. The Owner’s Virtual Machine performs the validation as in steps 1 (token security evaluation), 2 (token content validation) and 3 (token permissions evaluation) described above

#### -----------------------End of TS-0003 change 1----------------------------------------------

#### -----------------------Start of TS-0003 change 2---------------------------------------------

#### 7.3.2.6 oneM2M JSON Web Tokens (JWTs)

##### 7.3.2.6.1 Introduction to oneM2M JWTs

oneM2M specifies a JSON Web Tokens (JWTs) representation (IETF RFC 7519 [53]) for Tokens used in oneM2M. A JWT compliant with the present clause is called a *oneM2M JWT.*

**Background:** A JWT uses either the JSON Web Signature (JWS) Compact Representation, or JSON Web Encryption (JWE) Compact Representation, specified in IETF RFC 7515 [51] and IETF RFC 7519 [53]. The JWT specification IETF RFC 7519 [53] also defines an unsecured JWT which is a JWS using the "alg" Header Parameter value "none" and with the empty string for its JWS Signature value.

The JWT specification defines a JSON element which is the structure of the payload of the JWS or JWE when used as a JWT. This payload comprises a set of JWT claims, with IETF RFC 7519 [53] standardizing an initial set of JWT claim names. IANA maintains a registry of JWT claim names [i.18].

##### 7.3.2.6.2 oneM2M JWT Profile

**oneM2M JWT Claims:** Table 7.3.2.6.2-1 provides the mapping from the JWT claim names, in a oneM2M JWT, to the elements of the m2m:tokenClaimSet complex data type described in oneM2M TS-0004 [4]. Where available, JWT claim names registered with IANA [i.18] have been used. oneM2M TS-0004 [4] specifies which elements are mandatory and which elements are optional.

Table 7.3.2.6.2-1: The oneM2M JWT claim set and mapping to elements of m2m:tokenClaimSet

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Token Claimset Object Element Path | Token Claimset Object Element Short Name | oneM2M JWT claim name | Where is this JWT claim name defined? | Additional details for mapping from Token Claimset Object values to JWT Claim values |
| version | ***tkvr*** | "tkvr" | oneM2M TS-0004 [4] short names | Values shall be identical |
| tokenID | ***tkid*** | "jti" | IETF RFC 7519 [53] | Values shall be identical |
| issuer | ***tkis*** | "iss" | IETF RFC 7519 [53] | Values shall be identical |
| holder | ***tkhd*** | "azp" | OpenID Connect Core 1.0 [54] | Values shall be identical |
| notBefore | ***tknb*** | "nbf" | IETF RFC 7519 [53] | Token Claimset Object element "notBefore" is in ISO8601 "Basic Format", see [4]. This element shall be mapped to JWT Claim "nbf" which uses NumericDate format [53]. |
| notAfter | ***tkna*** | "exp" | IETF RFC 7519 [53] | Token Claimset Object element "notAfter" is in ISO8601 "Basic Format", see [4]. This element shall be mapped to JWT Claim "exp" which uses NumericDate format [53]. |
| tokenName | ***tknm*** | "tknm" | oneM2M TS-0004 [4] short names | Values shall be identical |
| audience | ***tkau*** | "aud" | IETF RFC 7519 [53] | Token Claimset Object element "audience" is a list of m2m:ID. This list shall be mapped to JWT Claim "aud" comprising an array of case-sensitive strings, each containing a StringOrURI value [53]. |
| permissions | ***tkps*** | "tkps" | oneM2M TS-0004 [4] short names | Values shall be identical |
| extension | ***tkex*** | "tkex" | oneM2M TS-0004 [4] short names | Values shall be identical |
| nestedToken | ***tkobj*** | “tkobj” | oneM2M TS-0004 [4] short names | Values shall be identical |

**oneM2M JWT Security Profile**. The JWS Compact Representation and JWE Compact Representation are both supported by ESData (see clause 8.5.3). A oneM2M JWT may use any ESData security class: Encryption-only, Signature-only or Nested-Sign-then-encrypt. A oneM2M JWT may use any algorithm supported by ESData for the JWS Compact Representation and JWE Compact Representation.

A oneM2M JWT may be an unsecured JWT, in which case the oneM2M JWT is considered to use the unsecured ESData security class.

IETF RFC 7519 [53] discusses security considerations of JWTs, and operators of Token Issuers (Dynamic Authorization Servers and Authorization Authorities) should consult that text when deciding on ESData security class and algorithms.

**JOSE header parameters of oneM2M JWTs.** When the Encryption-only ESData security class is used, then

* The JOSE header of the JWE shall include the "typ" parameters set to "JWT".
* The JOSE header of the JWE shall not include the "cty" parameter.

When the Signature -only ESData security class is used, then

* The JOSE header of the JWS shall include the "typ" parameters set to "JWT".
* The JOSE header of the JWS shall not include the "cty" parameter.

When the Nested-Sign-then-encrypt ESData security class is used, then the JWT claims are the payload of a JWS, and the JWS becomes the payload of a JWE. In this case,

* The JOSE header of both the JWS and the JWE shall include the "typ" header parameters set to "JWT".
* The JOSE header of the JWE shall include the "cty" parameter set to be "JWT", to indicate that a Nested JWT is carried in this JWT.
* The JOSE header of the JWS shall not include the "cty" parameter.

##### 7.3.2.6.3 oneM2M JWT Procedures

**Configuring CSEs for verifying Tokens from a Token Issuer.** In order for a CSE to verify oneM2M JWTs issued by a particular Token Issuer, the CSE shall be provided with the following information in a secure manner

* The combinations of ESData Security classes and algorithms permitted by the Token Issuer.
* Credentials for verifying Tokens conforming to those ESData Security classes and algorithms, noting that no credentials are needed for verifying tokens using the unsecured ESData Security class.

The present document does not specify mechanisms for providing this information to the CSE. The present document does not define data structures for storing this information on the CSE. The security level to apply on each particular CSE has to be derived from application specific risk assessment.

**Creating a oneM2M JWT**: When a Token Issuer is triggered to create a token, then the Token issuer shall perform the following steps

1. The Token Issuer shall form a Token Claimset Object compliant with data type m2m:tokenClaimSet, with the permission element using the JSON serialization.
2. The Token Issuer shall create the corresponding oneM2M JWT claim set using the mapping in   
   Table 7.3.2.6.2-1.
3. If necessary, the Token Issuer shall include a nested Token. This token is either a full JWT token or a reference to a JWT by its TokenID. This token can be emitted by a 3rd party or by the Token Issuer itself.
4. The Token Issuer shall select an ESData Security Class, algorithms and corresponding credentials. This step may also be performed before step 1) or between steps 1) and 2).
5. The Token Issuer shall create oneM2M JWT using the oneM2M JWT claims, ESData Security Class, algorithms and corresponding credentials. This step uses the process described for JWTs in IETF RFC 7519 [53].

The resulting oneM2M JWT has data type m2m:dynAuthJWT.

**Validating a oneM2M JWT**: When a CSE receives a oneM2M JWT for use in an access decision, then the CSE shall perform the following steps

1. The CSE shall validate that the oneM2M JWT conforms to the m2m:dynAuthJWT data type.
2. The CSE shall validate the security of the oneM2M JWT as described in clause 7.3.2.5, using the JWT-specific details in IETF RFC 7519 [53] and configured credentials (if required). A CSE shall discard a oneM2M JWT which uses a ESData Security class or algorithms which are not permitted by the Token Issuer.
3. The CSE shall create a Token Claimset Object from the oneM2M JWT claim set by reversing the mapping in Table 7.3.2.6.2-1.
4. The CSE shall validate the Token Claimset Object as described in clause 7.3.2.5.

The Token Claimset Object permissions element can now be processed as described in clause 7.3.2.5.

### -----------------------End of TS-0003 change 2---------------------------------------------

CHECK LIST

* Does this Change Request include an informative introduction containing the problem(s) being solved, and a summary list of proposals.?
* Does this CR contain changes related to only one particular issue/problem?
* Have any mirror CRs been posted?
* Does this Change Request make **all** the changes necessary to address the issue or problem? E.g. A change impacting 5 tables should not include a proposal to change only 3 tables?Does this Change Request follow the drafting rules?
* Are all pictures editable?
* Have you checked the spelling and grammar?
* Have you used change bars for all modifications?
* Does the change include the current and surrounding clauses to clearly show where a change is located and to provide technical context of the proposed change? (Additions of complete clauses need not show surrounding clauses as long as the proposed clause number clearly shows where the new clause is proposed to be located.)
* Are multiple changes in this CR clearly separated by horizontal lines with embedded text such as, start of change 1, end of change 1, start of new clause, end of new clause.?