TTC仕様書
Technical Specification

TS-M2M-0009v4.4.0

サービス層 API 仕様（HTTP 用）

HTTP Protocol Binding

2023 年 3 月 23 日制定

一般社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TS-M2M-0009v4.4.0

サービス層 API 仕様（HTTP 用）　[HTTP Protocol Binding]

＜参考＞ [Remarks]

１．英文記述の適用レベル [Application level of English description]

適用レベル [Application level]：E2

本標準の本文、付属資料および付録の文章および図に英文記述を含んでいる。

[English description is included in the text and figures of main body, annexes and appendices.]

２．国際勧告等の関連 [Relationship with international recommendations and standards]

本標準は、oneM2M で承認された Technical Specification TS-0009-V4.4.0 に準拠している。

[This standard is standardized based on the Technical Specification TS-0009-V4.4.0 approved by oneM2M.]

３．上記国際勧告等に対する追加項目等 [Departures from international recommendations]

原標準に対する変更項目 [Changes to original standard]

原標準が参照する標準のうち、TTC 標準に置き換える項目。[Standards referred to in the original standard, which are replaced by TTC standards.]

原標準が参照する標準のうち、それらに準拠した TTC 標準等が制定されている場合は自動的に最新版 TTC 標準等に置き換え参照するものとする。　[Standards referred to in the original standard should be replaced by derived TTC standards.]

４．工業所有権 [IPR]

本標準に関わる「工業所有権等の実施の権利に係る確認書」の提出状況は、ＴＴＣホームページによる。

[Status of “Confirmation of IPR Licensing Condition” submitted is provided in the TTC web site.]

５．作成専門委員会 [Working Group]

oneM2M 専門委員会 [oneM2M Working Group]

oneM2M
Technical Specification

Document Number	TS-0009-V4.4.0
Document Name:	HTTP Protocol Binding
Date:	2022-02-16
Abstract:	HTTP Protocol Binding TS

About oneM2M

The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide.

More information about oneM2M may be found at: http//www.oneM2M.org

Copyright Notification

Notice of Disclaimer & Limitation of Liability

Contents

1 Scope

The present document will cover the protocol specific part of communication protocol used by oneM2M compliant systems as RESTful HTTP binding.

The scope of the present document is (not limited to as shown below):

- Binding oneM2M Protocol primitive types to HTTP method.
- Binding oneM2M response status codes (successful/unsuccessful) to HTTP response codes.
- Binding oneM2M RESTful resources to HTTP resources.

The present document is depending on Core Protocol specification (oneM2M TS-0004) for data types.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

The following referenced documents are necessary for the application of the present document.

[1] IETF RFC 7230 (June 2014): "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing".

[2] oneM2M TS-0003: Security Solutions.

[3] oneM2M TS-0004: "Service Layer Core Protocol Specification".

[4] IETF RFC 7235 (June 2014): "Hypertext Transfer Protocol (HTTP/1.1): Authentication".

[5] IETF RFC 6750 (October 2012): "The Oauth 2.0 Authorization Framework: Bearer Token Usage".

[6] oneM2M TS-0011: Common Terminology.

[7] oneM2M TS-0001: Functional Architecture.

[8] IETF RFC 7232 (June 2014): "Hypertext Transfer Protocol (HTTP/1.1): "Conditional Requests".

[9] IETF RFC 3986 (January 2005): "Uniform Resource Identifier (URI): Generic Syntax".

[10] IETF RFC 7231 (June 2014): "HTTP/1.1 Semantics and Content"

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] oneM2M Drafting Rules.

NOTE: Available at http://www.onem2m.org/images/files/oneM2M-Drafting-Rules.pdf.

[i.2] Void

[i.3] Void

150 [i.4] IETF RFC 6455 (December 2011):"The WebSocket Protocol".

151 [i.5] Void

152 3 Abbreviations

153 For the purposes of the present document, the abbreviations given in TS-0011 [6] apply and the following apply:

154 HTTP Hyper Text Transfer Protocol
155 TLS Transport Layer Security
156 URI Uniform Resource Identifier

157 4 Conventions

158 The keywords "Shall", "Shall not", "May", "Need not", "Should", "Should not" in this document are to be interpreted
159 as described in the oneM2M Drafting Rules [i.1].

160 5 Overview on HTTP Binding

161 5.0 Overview

162 HTTP binding specifies the equivalence between oneM2M request and response primitives and HTTP request and
163 response messages, respectively. This clause provides a brief overview on the mapping relationship between oneM2M
164 and HTTP message parameters.

165 This clause describes how oneM2M request/response primitives can be mapped to HTTP request/response messages
166 and vice versa.

167 5.1 Introduction

168 Figure 5.1-1 illustrates an example oneM2M system configuration and its correspondence to an HTTP-based
169 information system if HTTP binding as defined in this specification is applied. The upper diagram in figure 5.1-1 shows
170 with solid line arrows the flow of a request primitive originating from an AE which is registered to an MN-CSE
171 (Registrar of AE). The request primitive is assumed to address a resource which is hosted by another MN-CSE (Host of
172 Resource). Both MN-CSEs are registered to the same IN-CSE.

173 When applying HTTP binding, the oneM2M entities of the upper diagram take the roles outlined in the lower diagram
174 of a corresponding HTTP information system as defined in IETF RFC 7230 [1]. The AE takes the role of an HTTP
175 client, the MN-CSE (Registrar of AE) takes the role of a HTTP Proxy Server, and both the IN-CSE and MN-CSE (Host
176 of Resource) take the role of a HTTP server for this particular request message.

177 CSEs may also issue unsolicited request messages, shown with dashed line arrows in figure 5.1-1, and receive
178 associated response messages. Therefore, for HTTP protocol binding, CSEs generally provides capability of both HTTP
179 Server and HTTP Client. Aes may provide HTTP Server capability optionally in order to be able to serve Notification
180 request messages (see TS-0004 [3] and TS-0001 [7]).

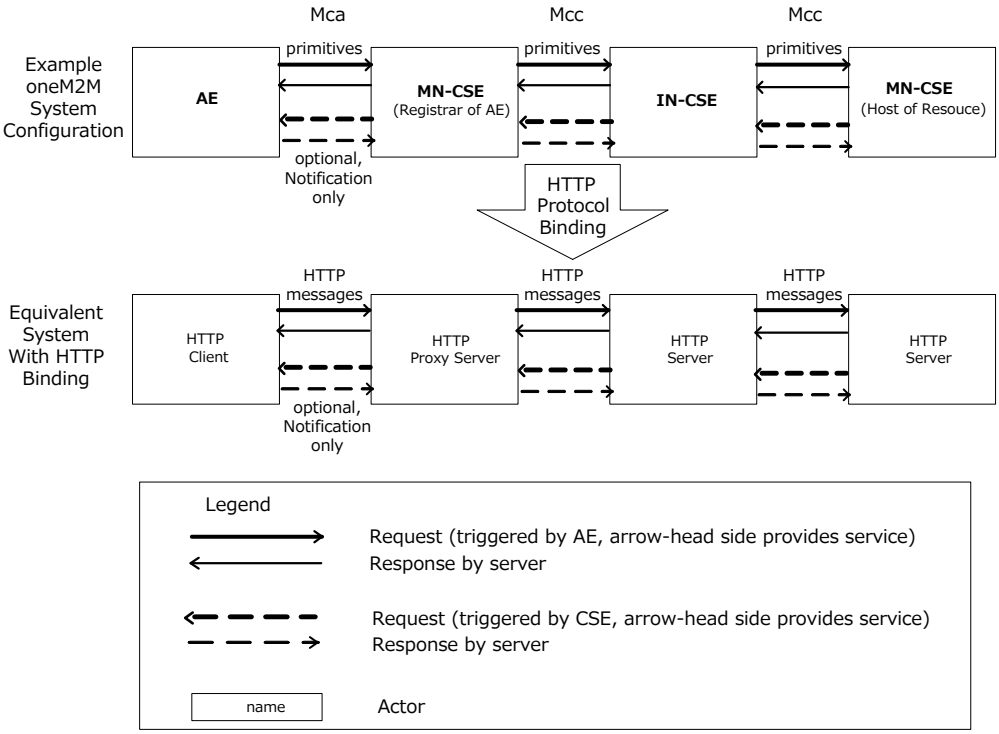

Figure 5.1-1 : Correspondence between oneM2M entities and HTTP Client and Server

Each individual request primitive will be mapped to single HTTP request message, and each individual response primitive will be mapped to a single HTTP response message, and vice-versa.

An HTTP request message consists of Request-Line, headers and message-body. An HTTP response message consists of Status-Line, headers and message-body [1]. HTTP header names are case-insensitive and a Receiver shall accept headers that are either lower or upper or any mixture thereof. This clause describes how oneM2M request/response primitives are mapped to HTTP messages at a high level. Corresponding details are specified in clause 6.

5.2 Request-Line

The HTTP method of a request message is mapped to the ***Operation*** parameter, and vice-versa.

At the message originator side the HTTP Request-Target is derived from the ***To*** parameter of the request primitive, including a query string which carries other specific primitive parameters.

HTTP-Version is specified in clause 6.

5.3 Status-Line

HTTP Version is specified in clause 6.

The Status-Code of HTTP response messages is derived from the ***Response Status Code*** parameter of the response primitive. The Reason-Phrase is not applicable to oneM2M systems and is omitted.

6 HTTP Message Mapping

6.1 Introduction

Mapping between oneM2M primitives and HTTP messages shall be applied in the following four use cases:

1) Mapping of request primitive to HTTP request message at the request originator (HTTP client)
2) Mapping of HTTP request message to request primitive at the request receiver (HTTP server)
3) Mapping of response primitive to HTTP response message at the request receiver (HTTP server)

4) Mapping of HTTP response message to response primitive at the request originator (HTTP client)

All four use cases also appear at transit CSEs.

The following clauses specify the mapping between each oneM2M primitive parameter and a corresponding HTTP message field to compose a HTTP request/response message.

6.2 Parameter Mappings on Request-Line

6.2.1 Method

The HTTP 'Method' shall be derived from the ***Operation*** request primitive parameter of the request primitive.

Table 6.2.1-1: HTTP Method Mapping

oneM2M Operation	HTTP Method
Create	POST
Retrieve	GET
Update	PUT
Delete	DELETE
Notify	POST

At the Receiver, an HTTP request message with POST method shall be mapped either to a Create or Notify ***Operation*** parameter. Discrimination between Create and Notify operations can be accomplished by inspection of the content-type header. The ***Resource Type*** parameter is present in the content-type header only when the HTTP POST request represents a Create request (see clause 6.4.3). The ***Resource Type*** parameter is not present in the content-type header when the HTTP POST request represents a Notify request.

6.2.2 Request-Target

6.2.2.1 Path component

The path component of the origin-form HTTP Request-Target shall be interpreted as the mapping of the resource identifier part of the ***To*** request primitive parameter. If the HTTP message is sent directly to the next hop CSE, the origin-form of Request-Target shall be employed (see clause 5.3.1 of IETF RFC 7230 [1]).

The resource identifier part of the ***To*** parameter can be represented in three different forms (see clause 6.2.3 of oneM2M TS-0004 [3] and clause 7.2 of oneM2M TS-0001 [7]):

- CSE-Relative-Resource-ID,
- SP-Relative-Resource-ID,
- Absolute-Resource-ID.

Each of the above three formats may include either a structured Resource ID (used for hierarchical addressing) or an unstructured Resource ID (used for non-hierarchical addressing) as defined in clause 7.2 of oneM2M TS-0001 [7].

For CSE-relative Resource ID representation, the path component of the HTTP request message shall be constructed as the concatenation of the literal "/" and the ***To*** request primitive parameter. For SP-relative Resource ID representation, the path component of the HTTP request message shall be constructed as the concatenation of the literal "/~" and the ***To*** request primitive parameter. For Absolute Resource ID representation, the path component of the HTTP request message shall be constructed by replacing the first "/" character of the ***To*** request primitive parameter with "/_".

Table 6.2.2.1-1 shows valid mappings between the ***To*** request primitive parameter and the path component of the origin-form HTTP request target. In the shown examples, /myCSEID and /CSE178 represent applicable CSI-IDs, CSEBase represents the resource name of a <CSEBase> resource, CSEBase/ae12/cont27/contInst696 represents a structured CSE-relative resource ID, and cin00856 an unstructured CSE-relative resource ID.

Table 6.2.2.1-1: Mapping examples between *To* parameter and path component of request-line

Resource-ID Type	*To* parameter value	path component (origin-form)
structured CSE-Relative	CSEBase/ae12/cont27/contInst696	/CSEBase/ae12/cont27/contInst696/
unstructured CSE-Relative	cin00856	/cin00856
structured SP-Relative	/CSE178/CSEBase/ae12/cont27/contInst696	/~/CSE178/CSEBase/ae12/cont27/contInst696
unstructured SP-Relative	/CSE178/cin00856	/~/CSE178/cin00856
structured Absolute	//mym2msp.org/CSE178/CSEBase/ae12/cont27/contInst696	/_/mym2msp.org/CSE178/CSEBase/ae12/cont27/contInst696
unstructured Absolute	//mym2msp.org/CSE178/cin00856	/_/mym2msp.org/CSE178/cin00856

At the HTTP server side, the reverse operations shall be applied to the path component of request-line to derive a replica of the original ***To*** request primitive parameter.

If the HTTP message is sent to a HTTP proxy instead directly to the next hop CSE, the absolute-form of Request-Target shall be employed (see clause 5.3.2 of IETF RFC 7230 [1]). The absolute-form is derived by prefixing the origin-form with the schema and the host address of the next hop CSE:

http://{host address of next hop CSE}{origin-form path-component}

6.2.2.2 Query component

The query component (e.g. query-string) may include the optional primitive parameters listed in table 6.2.2.2-1 compliant with IETF RFC 7230 [1]. Each applicable request primitive parameters and elements of ***Filter Criteria*** parameter shown in table 6.2.2.2-1 shall be represented as pair of field-name and value in query-string. Multiple such pairs shall be concatenated with an ampersand '&' character used as separator between two pairs.

Table 6.2.2.2-1 also shows the permitted multiplicity of occurrence of field names in the query-string. Multiplicity '0..1' means that a parameter is optional and can occur at most once. Parameters with multiplicity '0..n', may occur multiple times in the query-string in the form of <query field name> = value. For example, if the resourceType element of the ***Filter Criteria*** parameter is represented by a list of 3 values '2 3 4' (see clause 6.3.4.7 in TS-0004 [3]), it would be mapped to ty=2+3+4 in the query-string. At the receiver side, this query string can be reverted back into the list type of representation. The same representation shall be applied for multiple occurrences of contentType and labels elements.

The 'attribute' element of the ***Filter Criteria*** request primitive parameter consists of two elements, name and value, which in XML notation would look for example as follows in case of multiplicity 2 (see clause 6.2.4.8 in TS-0004 [3]):

```
<attribute>
   <name>attname1</name>
   <value>attvalue1</value>
</attribute>
<attribute>
  <name>attname2</name>
  <value>attvalue2</value>
</attribute>
```

Each name (e.g. attname1 and attname2) shall represent a valid resource attribute name of the resource types indicated in the ty field of the query-string. The sequence of attribute elements as shown in the above example will be mapped into the query-string as attname1=attvalue1&attname2=attvalue2. The attribute names (i.e. attname1 and attname2 in the above example) shall be expressed in the form of short names as defined in clause 8.2.3 of TS-0004 [3]. Note that the <attribute> tag of the XML representation is omitted in the HTTP binding.

The 'childAttribute' and 'parentAttribute' elements of the ***Filter Criteria*** request primitive are handled in a similar way to the 'attribute' element. Those sequences of attribute elements will be mapped in the query string by adding a prefix to each attribute name respectively: 'c.' for 'childAttribute' and 'p.' for 'parentAttribute'. This results, using the example

above, in the mappings as c.attname1=attvalue1&c.attname2=attvalue2 for 'childAttribute', and p.attname1=attvalue1&p.attname2=attvalue2 for 'parentAttribute'.

Examples of valid Request-Target representations are the following:

EXAMPLE 1): Request-Target for 'nonBlockingRequestSynch'

Primitive parameters: To: /CSE1234/RCSE78/container234 (SP-Relative-Resource-ID)

Response Type: responseType = 1 (nonBlockingRequestSynch)

Result Persistence: P1Y2M3DT10H1M0S Request-Target: /CSE1234/RCSE78/container234?rt=1&rp=P1Y2M3DT10H1M0S

EXAMPLE 2): Request-Target for Discovery

When the entity wants to discover container resources where the *creator* attribute has the value 'Sam':

Primitive parameters: To: /CSE1234/RCSE78

Filter Criteria: resourceType = 3 (container)

attribute name: creator

attribute value: Sam

filterUsage = discovery

Request-Target: /CSE1234/RCSE78?ty=3&cr=Sam&fu=1

EXAMPLE 3): Semantic Discovery

The entity wants to discover resources whose semantic description stored in the *descriptor* attribute of a <semanticDescriptor> child resource fulfils the semantic filter specified in SPARQL. In this case, the semantic descriptor of the resource to discover has to contain information about a Thing of type Car based on the concept defined in the "myOnt" ontology.

Due to the use of reserved characters in SPARQL, the semanticsFilter requires "percent-encoding" [9].

Primitive parameters: To: /CSE1234/RCSE78

Filter Criteria: semanticsFilter =
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
PREFIX myOnt: http://www.onem2m.org/ontology/myontology#
SELECT ?car WHERE { ?car rdf:type myOnt:Car }

Request-Target: /CSE1234/RCSE78?smf=PREFIX%20rdf%3A%20%3Chttp%3A%2F%2Fwww.w3.org%2F1999%2F02%2F22-rdf-syntax-ns%23%3E%20PREFIX%20myOnt%3A%20%3Chttp%3A%2F%2Fwww.onem2m.org%2Fontology%2Fmyontology%23%3E%20SELECT%20%3Fcar%20WHERE%20%7B%20%3Fcar%20%20rdf%3Atype%20myOnt%3Acar%20%7D

EXAMPLE 4): Geo-query

When an application wants to query a resource having its geo-location within the rectangle:

Primitive parameters: To: /CSE1234/RCSE78

Filter Criteria: geometryType = 3 (Polygon)

geometry = [[0.0, 0.0], [0.0, 100.0], [100.0, 100.0], [100.0, 0.0], [0.0, 0.0]]

geoSpatialFunction = 1 (Within)

filterUsage = discovery

Request-Target:

/CSE1234/RCSE78?fu=1&gmty=3&geom=[[0.0,0.0],[0.0,100.0],[100.0,100.0],[100.0,0.0],[0.0,0.0]]&gsf=1

Note that, in the HTTP Request-Target, longitude and latitude are separated by an empty space and each pair of longitude and latitude are separated by comma.

Any of the short names listed in table 6.2.2.2-1, with the exception of 'atr', may be used in the query-string. The short name 'atr' itself is not used. Instead, any of the resource attribute short names as listed in tables 8.2.3-1 to 8.2.3-5 in oneM2M TS-0004 [3] may be used in the query-string in representations of attname=attvalue expressions, except those that shall be omitted (see clause 7.3.3.17.9 in oneM2M TS-0004 [3]).

Table 6.2.2.2-1: oneM2M request parameters mapped as query-string field

Request Primitive Parameter	Query Field Name	Multiplicity	Note
Response Type	***rt***	0..1	*responseType* element of data type ***responseTypeInfo*** (cf. clause 6.3.4.29 of TS-0004 [3])
Result Persistence	***rp***	0..1	
Result Content	***rcn***	0..1	
Delivery Aggregation	***da***	0..1	Allowed values are boolean-typed parameter. 'false', 'true', '0' and '1'
createdBefore	***crb***	0..1	filterCriteria condition
createdAfter	***cra***	0..1	filterCriteria condition
modifiedSince	***ms***	0..1	filterCriteria condition
unmodifiedSince	***us***	0..1	filterCriteria condition
stateTagSmaller	***sts***	0..1	filterCriteria condition
stateTagBigger	***stb***	0..1	filterCriteria condition
expireBefore	***exb***	0..1	filterCriteria condition
expireAfter	***exa***	0..1	filterCriteria condition
labels	***lbl***	0..n	filterCriteria condition
resourceType	***ty***	0..n	filterCriteria condition
sizeAbove	***sza***	0..1	filterCriteria condition
sizeBelow	***szb***	0..1	filterCriteria condition
contentType	***cty***	0..n	filterCriteria condition
limit	***lim***	0..1	filterCriteria condition
attribute	***atr***	0..n	filterCriteria condition
filterUsage	***fu***	0..1	filterCriteria condition
semanticsFilter	***smf***	0..n	filterCriteria condition, shall use "percent-encoding" [9] where required, see example 3)
filterOperation	***fo***	0..1	filterCriteria condition
contentFilterSyntax	***cfs***	0..1	filterCriteria condition
contentFilterQuery	***cfq***	0..1	filterCriteria condition
level	***lvl***	0..1	filterCriteria condition
offset	***ofst***	0..1	filterCriteria condition
geometryType	***gmty***	0..1	filterCriteria condition
geometry	***geom***	0..1	filterCriteria condition
geoSpatialFunction	***gsf***	0..1	filterCriteria condition
Discovery Result Type	***drt***	0..1	
Role IDs	***rids***	0..n	
Token IDs	***tids***	0..n	
LocalTokenIDs	***ltids***	0..n	
Token Request Indicator	***tqi***	0..n	Allowed values are boolean-typed parameter. 'false', 'true', '0' and '1'
Authorization Signature Indicator	***asi***	0..1	Allowed values are boolean-typed parameter. 'false', 'true', '0' and '1'
Authorization Relationship Indicator	***auri***	0..1	Allowed values are boolean-typed parameter. 'false', 'true', '0' and '1'
Semantic Query Indicator	***sqi***	0..1	Allowed values are boolean-typed parameter. 'false', 'true', '0' and '1'

For partial Retrieve request primitives, the ***To*** parameter may include the name of a single attribute separated by a '#' character from the resource ID. If multiple resource attributes are to be retrieved with a partial retrieve request

primitive, these attributes are included in form of an attributeList object (as specified in Table 6.3.3-1 of TS-0004 [3]) in the ***Content*** parameter.

In both cases, the short resource attribute name(s) shall be included into the fragment component of request-target, i.e. it shall follow any required query-string separated by '#' character. If more than a single attribute name is included into the fragment component, these shall be separated by a '+' character.

For example, if three resource attributes with long names resourceID, labels and requestReachability are indicated in the ***Content*** primitive parameter, the query component atrl=ri+lbl+rr is attached to the request-target. In case just a single attribute "rr" is indicated in the ***To*** parameter separated by '#' character, the query component atrl=rr is attached to the request-target. The '#' character and following attribute name shall be omitted from the path component of the request line.

Case 1 Primitive Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:rqp xmlns:m2m="http://www.onem2m.org/xml/protocols">
   <op>1</op>
   <to>//example.net/myCSE/-/Cont1</to>
   <fr>/myCSE/C2345</fr>
   <rqi>0002bf63</rqi>
   <ty>4</ty>
   <pc>
      <atrl>ri lbl rr</atrl>
   </pc>
</m2m:rqp>
```

Case 1 HTTP Binding: **//example.net/myCSE/-/Cont1?atrl=ri+lbl+rr**

Case 2 Primitive Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:rqp xmlns:m2m="http://www.onem2m.org/xml/protocols">
   <op>2</op>
   <to>//example.net/myCSE/-/Cont1#rr</to>
   <fr>/myCSE/C2345</fr>
   <rqi>0002bf63</rqi>
   <ty>4</ty>
</m2m:rqp>
```

Case 2 HTTP Binding: **//example.net/myCSE/-/Cont1?atrl=rr**

At the HTTP server side, the reverse operation shall take place, when constructing the retrieve request primitive from the receive HTTP request message. Single attribute names in the query component may either be mapped back into the ***To*** parameter following a '#' character, or included into the ***Content*** parameter using the attributeList format with just a single list element included. Multiple attributes shall be included into the ***Content*** parameter as specified in oneM2M TS-0004 [3].

6.2.3 HTTP-Version

This specification defines binding compliant with HTTP 1.1 [1]. The HTTP version field in HTTP request messages shall be set to "HTTP/1.1".

6.3 Status-Line

6.3.1 HTTP-Version

The HTTP version field in HTTP response messages shall be set to “HTTP/1.1”.

6.3.2 Status-Code

The ***Response Status Code*** parameter of response primitives shall be mapped to the HTTP Status-Code. Since the ***Response Status Code*** parameter values have been defined with more detailed information than HTTP status codes, one or more ***Response Status Code*** value may be mapped to the same HTTP Status-Code. The original ***Response Status Code*** parameter value shall be carried in the X-M2M-RSC header (see clause 6.4.17).

The mapping of ***Response Status Code*** parameter value of oneM2M request primitive to Status-Code of HTTP request messages is specified in table 6.3.2-1.

Table 6.3.2-1: Status Code MappingoneM2M Response Status Codes	HTTP Status Codes [10]
2000 (OK)	200 (OK)
2002 (DELETED)	
2004 (UPDATED)	
2001 (CREATED)	201 (Created)
1000 (ACCEPTED)	202 (Accepted)
1001 (ACCEPTED for nonBlockingRequestSynch)	
1002 (ACCEPTED for nonBlockingRequestAsynch)	
4000 (BAD_REQUEST)	400 (Bad Request)
4102 (CONTENTS_UNACCEPTABLE)	
4110 (GROUP_MEMBER_TYPE_INCONSISTENT)	
4120 (INVALID_SEMANTICS)	
4122 (INVALID_TRIGGER_PURPOSE)	
4123 (ILLEGAL_TRANSACTION_STATE_TRANSITION_ATTEMPTED)	
4131 (ONTOLOGY_MAPPING_POLICY_NOT_MATCHED)	
4133 (BAD_FACT_INPUTS_FOR_REASONING)	
4134 (BAD_RULE_INPUTS_FOR_REASONING)	
4137 (PRIMITIVE_PROFILE_BAD_REQUEST)	
4143 (INVALID_SPARQL_QUERY)	
6010 (MAX_NUMBER_OF_MEMBER_EXCEEDED)	
6022 (INVALID_CMDTYPE)	
6023 (INVALID_ARGUMENTS)	
6024 (INSUFFICIENT_ARGUMENTS)	
4101 (SUBSCRIPTION_CREATOR_HAS_NO_PRIVILEGE)	403 (Forbidden)
4103 (ORIGINATOR_HAS_NO_PRIVILEGE)	
5105 (RECEIVER_HAS_NO_PRIVILEGE)	
5203 (TARGET_NOT_SUBSCRIBABLE)	
5205 (SUBSCRIPTION_HOST_HAS_NO_PRIVILEGE)	
4106 (ORIGINATOR_HAS_NOT_REGISTERED)	
4107 (SECURITY_ASSOCIATION_REQUIRED)	
4108 (INVALID_CHILD_RESOURCE_TYPE)	
4109 (NO_MEMBERS)	
4111 (ESPRIM_UNSUPPORTED_OPTION)	
4112 (ESPRIM_UNKNOWN_KEY_ID)	
4113 (ESPRIM_UNKNOWN_ORIG_RAND_ID)	
4114 (ESPRIM_UNKNOWN_RECV_RAND_ID)	
4115 (ESPRIM_BAD_MAC)	
4116 (ESPRIM_IMPERSONATION_ERROR)	
4117 (ORIGINATOR_HAS_ALREADY_REGISTERED)	
4126 (APP_RULE_VALIDATION_FAILED)	
4127 (OPERATION_DENIED_BY_REMOTE_ENTITY)	
4128 (SERVICE_SUBSCRIPTION_NOT_ESTABLISHED)	
4135 (DISCOVERY_LIMIT_EXCEEDED)	
4136 (PRIMITIVE_PROFILE_NOT_ACCESSIBLE)	
4138 (UNAUTHORIZED_USER)	
4139 (SERVICE_SUBSCRIPTION_NOT_ACTIVE)	
5208 (DISCOVERY_DENIED_BY_IPE)	
5214 (TARGET_HAS_NO_SESSION_CAPABILITY)	
5215 (SESSION_IS_ONLINE)	
5218 (TRIGGERING_DISABLED_FOR_RECIPIENT)	
5222 (TRANSACTION_PROCESSING_IS_INCOMPLETE)	
6034 (REQUESTED_ACTIVITY_PATTERN_NOT_PERMITTED)	
4004 (NOT_FOUND)	404 (Not Found)
4118 (ONTOLOGY_NOT_AVAILABLE)	
4119 (LINKED_SEMANTICS_NOT_AVAILABLE)	
4121 (MASHUP_MEMBER_NOT_FOUND)	
4130 (ONTOLOGY_MAPPING_ALGORITHM_NOT_AVAILABLE)	
4132 (ONTOLOGY_MAPPING_NOT_AVAILABLE)	
5103 (TARGET_NOT_REACHABLE)	
5107 (REMOTE_ENTITY_NOT_REACHABLE)	
6003 (EXTERNAL_OBJECT_NOT_REACHABLE)	
6005 (EXTERNAL_OBJECT_NOT_FOUND)	
4005 (OPERATION_NOT_ALLOWED)	405 (Method Not Allowed)
5207 (NOT_ACCEPTABLE)	406 (Not Acceptable)
4104 (GROUP_REQUEST_IDENTIFIER_EXISTS)	409 (Conflict)

4105 (CONFLICT)	
4124 (BLOCKING_SUBSCRIPTION_ALREADY_EXISTS)	
4140 (SOFTWARE_CAMPAIGN_CONFLICT)	
5106 (ALREADY_EXISTS)	
5219 (UNABLE_TO_REPLACE_REQUEST)	
5220 (UNABLE_TO_RECALL_REQUEST)	
6028 (ALREADY_COMPLETE)	
6029 (MGMT_COMMAND_NOT_CANCELLABLE)	
4015 (UNSUPPORTED_MEDIA_TYPE)	415 (Unsupported Media Type)
5000 (INTERNAL_SERVER_ERROR)	500 (Internal Server Error)
5204 (SUBSCRIPTION_VERIFICATION_INITIATION_FAILED)	
5209 (GROUP_MEMBERS_NOT_RESPONDED)	
5210 (ESPRIM_DECRYPTION_ERROR)	
5211 (ESPRIM_ENCRYPTION_ERROR)	
5212 (SPARQL_UPDATE_ERROR)	
5216 (JOIN_MULTICAST_GROUP_FAILED)	
5217 (LEAVE_MULTICAST_GROUP_FAILED)	
5221 (CROSS_RESOURCE_OPERATION_FAILURE)	
5230 (ONTOLOGY_MAPPING_ALGORITHM_FAILED)	
5231 (ONTOLOGY_CONVERSION_FAILED)	
5232 (REASONING_PROCESSING_FAILED)	
6020 (MGMT_SESSION_CANNOT_BE_ESTABLISHED)	
6021 (MGMT_SESSION_ESTABLISHMENT_TIMEOUT)	
6025 (MGMT_CONVERSION_ERROR)	
6026 (MGMT_CANCELLATION_FAILED)	
6033 (NETWORK_QOS_CONFIG_ERROR)	
4001 (RELEASE_VERSION_NOT_SUPPORTED)	501 (Not Implemented)
4125 (SPECIALIZATION_SCHEMA_NOT_FOUND)	
5001 (NOT_IMPLEMENTED)	
5206 (NON_BLOCKING_SYNCH_REQUEST_NOT_SUPPORTED)	
4008 (REQUEST_TIMEOUT)	504 (Gateway Timeout)
6030 (EXTERNAL_OBJECT_NOT_REACHABLE_BEFORE_RQET_TIMEOUT)	
6031 (EXTERNAL_OBJECT_NOT_REACHABLE_BEFORE_OET_TIMEOUT)	

6.3.3 Reason-Phrase

The Reason-Phrase shall be omitted in HTTP response messages.

6.4 Header Fields

6.4.0 Introduction

The header fields listed in this clause shall be supported by all entities of the oneM2M system when using HTTP binding. Any other unrecognized HTTP headers shall be ignored by the HTTP client and server.

6.4.1 Host

The Host header shall be present in each HTTP request message.

While the Request-Target indicates a target resource on the Hosting CSE, the Host header indicates the FQDN or IP address of the Receiver CSE of the next hop in multi-hop communication scenarios. Therefore, the Request-Target is not changed but the Host header is changed each time when a request is forwarded to the next hop CSE.

When no HTTP proxy is used, the Host header shall be set as one of the pointOfAccess attribute values of the Receiver(i.e. pointOfAccess attribute of the corresponding <remoteCSE> resource). Selection of the appropriate Receiver is described in oneM2M TS-0004 [3]. In this case the origin-form of target URI shall be used (see clause 6.2.2).

If the HTTP request message is sent to a HTTP proxy rather than to the next hop CSE, the Host header shall be set to the FQDN or IP address of the proxy. In this case the absolute-form of target URI shall be used (see clause 6.2.2).

6.4.2 Accept

The Originator may use the Accept header to indicate which media types are acceptable for the response. The Accept header shall be mapped to a set of media types among “application/xml”, “application/json”, “application/cbor” or the oneM2M defined media types defined in clause 6.7 of oneM2M TS-0004 [3]. Note that some of the oneM2M defined media types defined in clause 6.7 of oneM2M TS-0004 [3] are not applicable for the response. Note that this information is not included in a request primitive.

6.4.3 Content-Type

Any HTTP request or response containing message-body shall include the Content-type header set to one of “application/xml”, “application/json”, or the oneM2M defined media types defined in clause 6.7 of oneM2M TS-0004 [3].

Content-Type of the HTTP response should be chosen by the Hosting CSE considering the Accept header given in the HTTP request.

The value of the Resource Type primitive parameter, which is present in Create request primitives only, shall be appended to the Content-type of the corresponding HTTP request message in the form ty=value, separated by a semicolon character. A valid Content-Type header in this case looks e.g. as follows:

Content-Type: application/vnd.onem2m-res+xml; ty=3

application/vnd.onem2m-res+json; ty=3

application/vnd.onem2m-res+cbor; ty=3

6.4.4 Content-Location

The Content-Location header of the HTTP response to a Create request shall be set to the URI of the created resource, if this URI is present in the ***Content*** parameter of the Create response primitive. See clause 7.3.3.12 “Create a success response” in oneM2M TS-0004 [3].

6.4.5 Content-Length

If message-body is included into HTTP request or response messages, the Content-Length header shall be included indicating the length of the message-body in octets (8-bit bytes).

6.4.6 Etag

A response primitive sent in reply to a resource retrieval request primitive should include an Etag header [8] in combination with the resource representation in the HTTP message body.

Etag facilitates the use of conditional requests (i.e. using the if-match and if-none-match HTTP headers) [8].

If a CSE supports the Etag header, then the CSE shall support conditional requests compliant with IETF RFC 7232 [8].

6.4.7 X-M2M-Origin

The X-M2M-Origin header shall be mapped to the ***From*** parameter of request and response primitives and vice versa, if applicable.

6.4.8 X-M2M-RI

The X-M2M-RI header shall be mapped to the ***Request Identifier*** parameter of request and response primitives and vice versa.

6.4.9 Void

6.4.10 X-M2M-GID

The X-M2M-GID header shall be mapped to the ***Group Request Identifier*** parameter of request primitives and vice versa, if applicable.

6.4.11 X-M2M-RTU

The X-M2M-RTU header shall be mapped to the *notificationURI* element of the ***Response Type*** parameter of request primitives and vice versa, if applicable. If there are more than one value in the element, then the values shall be combined with "&" character.

6.4.12 X-M2M-OT

The X-M2M-OT header shall be mapped to the ***Originating Timestamp*** parameter of request and response primitives, and vice versa, if applicable.

6.4.13 X-M2M-RST

The X-M2M-RST header shall be mapped to the ***Result Expiration Timestamp*** parameter of request and response primitives, and vice versa, if applicable.

6.4.14 X-M2M-RET

The X-M2M-RET header shall be mapped to the ***Request Expiration Timestamp*** parameter of request primitives and vice versa, if applicable.

6.4.15 X-M2M-OET

The X-M2M-OET header shall be mapped to the ***Operation Execution Time*** parameter of request primitives and vice versa, if applicable.

6.4.16 X-M2M-EC

The X-M2M-EC header shall be mapped to the ***Event Category*** parameter of request and response primitives, and vice versa, if applicable.

6.4.17 X-M2M-RSC

The X-M2M-RSC header in a HTTP response message shall be mapped to the ***Response Status Code*** parameter of response primitives and vice versa (e.g. ***Response Status Code*** 4000 and 4102 are mapped to HTTP Status Code 400 in the table 6.3.2-1).

6.4.18 X-M2M-ATI

The X-M2M-ATI header in a HTTP response message shall be mapped to the ***Assigned Token Identifiers*** parameter of response primitives and vice versa.

The format of the X-M2M-ATI header shall be represented as a sequence of lti-value:tkid-value pairs separated by a colon ":' and multiple pairs separated by '+' character.

EXAMPLE: The header looks as follows:

X-M2M-ATI: lti-value1:tkid-value1 + lti-value2:tkid-value2 + …

if the XML representation of the ***Assigned Token Identifiers*** parameter is given as (using short element names):

```
<ati>
    <ltia>
        <lti>lti-value1</lti>
        <tkid>tkid-value1</tkid>
    </ltia>
    <ltia>
       <lti>lti-value2</lti>
       <tkid>tkid-value2</tkid>
    </ltia>
    …
 </ati>
```

The data type m2m:dynAuthlocalTokenIdAssignments of the ***Assigned Token Identifiers*** parameter is defined in clause 6.3.5.43 of TS-0004 [3].

6.4.19 Authorization

If a request primitive includes a ***Tokens*** parameter it shall be mapped to the Authorization header.

The ***Tokens*** primitive parameter is represented as a space separated list of JSON Web Signature (JWS) and JSON Web Encryption (JWE) strings in Compact Serialization format of datatype m2m:dynAuthJWT as defined in clause 6.3.3 of TS-0004 [3].

When mapped into the Authorization header, each individual token in the ***Tokens*** primitive parameter shall be separated by '+' character.

For example, if the ***Tokens*** parameter consists of a list of two JWS/JWE Tokens,

eyJ0eXAiOiJK.eyJpc3MiOiJqb2UiLA0KIC.dBjftJeZ4CVP
eyJ0eXAiOiJK.eyJpc3MiOiJqb2UiLA0KIC.dBjftJeZ4CVP.5eym8TW_c8SuK.SdiwkIr3a.XFBoMYUZo

the Authorization header looks as follows:

Authorization: eyJ0eXAiOiJK.eyJpc3MiOiJqb2UiLA0KIC.dBjftJeZ4CVP+ eyJ0eXAiOiJK.eyJpc3MiOiJqb2UiL
A0KIC.dBjftJeZ4CVP.5eym8TW_c8SuK.SdiwkIr3a.XFBoMYUZo

The line break in the above example is for illustrative purposes and shall not be included into the Authorization header.

6.4.20 X-M2M-CTS

The X-M2M-CTS header shall be mapped to the ***Content Status*** parameter of response primitives and vice versa, if applicable.

6.4.21 X-M2M-CTO

The X-M2M-CTO header shall be mapped to the ***Content Offset*** parameter of response primitives, and vice versa, if applicable.

6.4.22 X-M2M-RVI

The X-M2M-RVI header shall be mapped to the ***Release Version Indicator*** parameter of request and response primitives, and vice versa.

6.4.23 X-M2M-VSI

The X-M2M-VSI header shall be mapped to the ***Vendor Information*** parameter of request and response primitives, and vice versa.

6.4.24 X-M2M-AS

If a request primitive includes an ***Authorization Signature*** parameter, it shall be mapped to the X-M2M-AS header.

The ***Authorization Signature*** primitive parameter is represented as a space separated list of URL-safe base64 encoded (base64url)) strings of datatype m2m:signatureList as defined in clause 6.3.3 of TS-0004 [3].

When mapped into the X-M2M-AS header, each individual signature in the ***Authorization Signature*** primitive parameter shall be separated by '+' character.

For example, if the ***Authorization Signature*** parameter consists of a list of two elements,

i6watmQQQ1y3GB-VsWq5fJKzQcBB4jRfH1bfJFj0JtFVtLotttzYyA==
IWijxQjUrcXBYoCei4QxjWo9Kg8D3p9tlWoT4t0_gyTE96639In0FZFY2_rvP-_bMJ01EArmKZsR5VW3rwoPxw==

the X-M2M-AS header looks as follows:

X-M2M-AS: i6watmQQQ1y3GB-VsWq5fJKzQcBB4jRfH1bfJFj0JtFVtLotttzYyA==+
IWijxQjUrcXBYoCei4QxjWo9Kg8D3p9tlWoT4t0_gyTE96639In0FZFY2_rvP-_bMJ01EarmKZsR5VW3rwoPxw==

The line break in the above example is for illustrative purposes and shall not be included into the X-M2M-AS header. Whitespace characters are insignificant and should not be included into the header.

6.4.25 X-M2M-ASRI

The X-M2M-ASRI header in a HTTP response message shall be mapped to the ***Authorization Signature Request Information*** parameter of response primitives and vice versa, if applicable.

6.4.26 X-M2M-OMR

The X-M2M-OMR header shall be mapped to the ***Ontology Mapping Resources*** parameter of request primitives and vice versa, if applicable. The format of the oneM2M-OMR header shall be represented as a sequence of oneM2M resource identifiers separated by a '+'.

EXAMPLE: The header looks as follows:

oneM2M-OMR: /IN-CSE-0001/omr1+/IN-CSE-0001/omr2+…

6.4.27 X-M2M-MSU

The X-M2M-MSU header shall be mapped to the ***M2M Service User*** parameter of request and response primitives, and vice versa, if applicable.

6.4.28 X-M2M-PRPI

The X-M2M-PRPI header shall be mapped to the ***Primitive Profile Identifier*** parameter of request primitives, and vice versa.

6.5 Message-body

Message-body shall be mapped to the ***Content*** parameter of request and response primitives, and vice versa, if applicable. This applies to the ***Content*** parameter of all primitives with the following exceptions:

1) For partial Retrieve request primitives. Attributes contained in the Content parameter of Retrieve request primitive shall be mapped to the fragment component of request-target, as specified in clause 6.2.2.2, and vice versa.

2) A ***Token Request Information*** parameter included in a response primitive shall be mapped into the message-body either as a XML or JSON serialized object. The Content-Type and Content-Length headers shall be set compliant with the data representation (i.e. Content-Type: application/xml or application/json depending on the serialization format). Note that the ***Token Request Information*** parameter is used in oneM2M error response primitives (X-M2M-RSC: 4103 “ORIGINATOR_HAS_NO_PRIVILEGE”) only, which do not carry any other primitive content.

   Error response messages which include the ***Token Request Information*** parameter in the Message-Body shall not include any debugging information.

6.6 Message Routing

HTTP request and response message routing shall be performed as described in HTTP/1.1 [1].

7 Security Consideration

7.1 Authentication on HTTP Request Message

When sending the credential to be checked by the Registrar CSE, Proxy-Authorization header should be used as specified in HTTP/1.1 (see IETF RFC 7235 [4]).

When sending the credential to be checked by Hosting CSE, Authorization header should be used as specified in HTTP/1.1 [4].

When the credential to be checked by Hosting CSE is an Access Token which is compatible with Oauth 2.0 framework (see IETF RFC 6750 [5]), the Bearer authentication scheme shall be used as specified in Oauth 2.0 framework.

NOTE: The oneM2M Security Solutions [2] does not provide any details on usage or provisioning of the token.

7.2 Transport Layer Security

oneM2M primitive parameters contained in HTTP messages may be protected by TLS in a hop-by-hop manner. For the details, see the oneM2M Security Solutions specification [2].

NOTE: Some provisioning schemes of oneM2M TS-0003 [2] enable the provisioning of end-to-end credentials, but protocols to establish security associations between non-adjacent nodes are not addressed by oneM2M in the present document.

Annex A (informative): Example Procedures

A.1 <container> resource creation

Figure A.1-1 is HTTP mapping of procedure described in clause 7.4.6.2.1 [3]. Note the example shown in the figure applies under the following assumptions:

- "CSE1" is the name (i.e. value of the resourceName attribute) of the <CSEBase> resource of the registrar CSE
- "cont1" is the name of the created <container> resource chosen by the registrar CSE

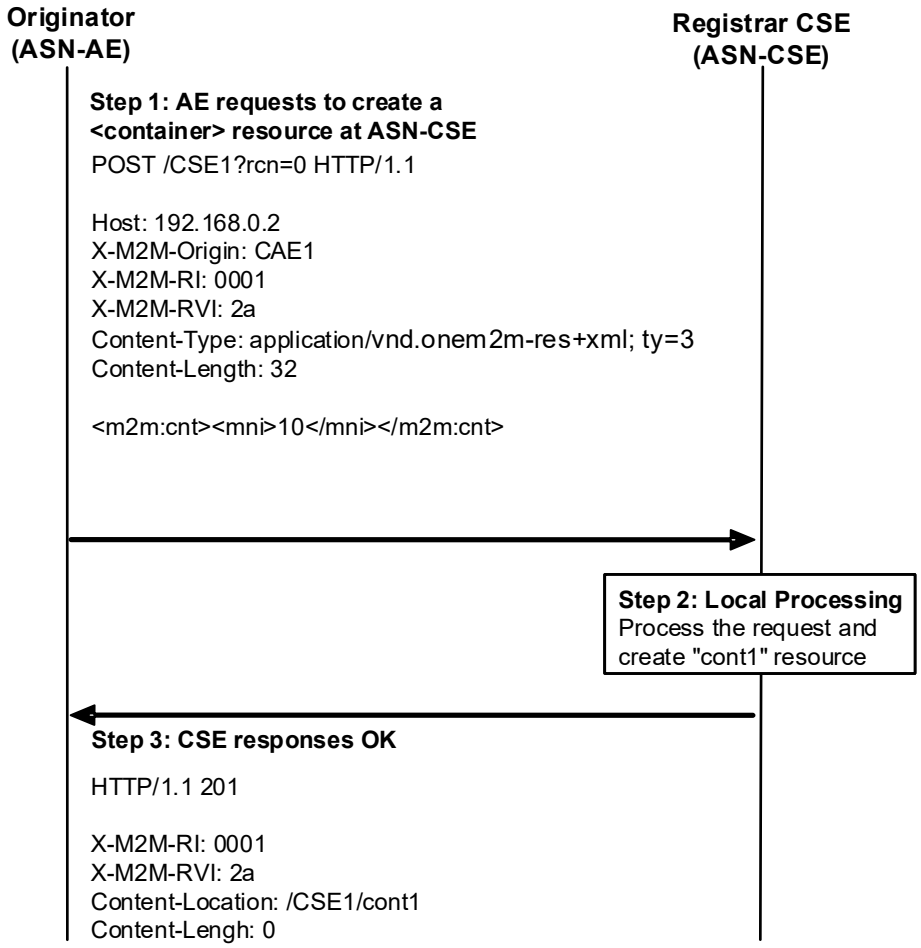

Figure A.1-1: oneM2M HTTP Binding Example – container creation

Annex B (informative): WebSocket

B.1 Notification using WebSocket

WebSocket [i.4] can be used for transporting notifications to an AE/CSE. This can be useful for an AE/CSE which is not server-capable or cannot be reachable for delivery of unsolicited requests.

For example, when an AE needs to receive a notification message from the CSE, the AE establishes a WebSocket connection to a CSE. When a new notification message is generated, the notification will be sent to the AE as the data frame of the WebSocket.

History

Publication history		

Draft history (to be removed on publication)		
v4.0.0	2020-04-22	V4.0.0 baseline is copied from v3.5.0 Also includes agreed contribution at SDS#43 meeting: 1. SDS-2019-0515R01-TS-0009_mapping_for_geo-query_filter_conditions 2. SDS-2019-0587R02-HTTP_Status_Code_for_ontology_mapping
v4.1.0	2021-01-18	Includes agreed contributions at SDS#48 meeting: 1. SDS-2020-0353-Adding_missing_URL_encodings_for_parent_and_child_attributes_to_TS-000 2. SDS-2020-0355-Correcting_wrong_reference_to_X-M2M-RSC 3. SDS-2020-0358R01-Removing_optimization_for_X-M2M-RSC_header Editor's correction on the Annex A.1 - Correct the reference number with the missing reference document info.
v4.2.0	2021-03-24	Includes agreed contributions at SDS#49 meeting: 1. SDS-2020-0380_Adding missing header mapping for M2M Service User Identifier 2. SDS-2020-0379_Corrected clause numbering in TS-0009 3. SDS-2020-0102R06_TS-0009_Missing_HTTP_Status_Codes_R4
v4.3.0	2021-12-01	Includes agreed contribution at SDS#51.x meeting: 1. SDS-2021-0193R01-TS-0009_Missing_HTTP_RSCs_Req_Params_R4
v4.4.0	2022-02-16	Includes agreed contribution at SDS#52 meeting: 1. SDS-2021-0252R01-Semantic_Query_Indicator_parameter_clarification 2. SDS-2021-0265R01-TS-0009_SPARQL_HTTP_Status_Codes_R4

600