TM Forum Specification

TMF630 REST API Design Guidelines Part 3

Guidelines for extending TMF Open APIâ•Žs with hypermedia support

TMF630

Team Approved Date: Nov. 25th, 2020

Release Status: Pre-Production	**Approval Status:** Team Approved
Version: 4.0.0	**IPR Mode:** RAND

Table of Contents

Executive Summary

This document, “REST API Design Guidelines Part 3” provides information for the development of hypermedia extensions to TM Forum APIs without payload modification.

It also includes guidelines on how JSON-LD support may also be added to the existing TMF Open API without payload modifications.

JSON-LD support in TMF Open API’s is optional.

An API can support hypermedia extension and can also be JSON-LD enabled.

Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Chapter 1. Hypermedia API and API State machine

The term Hypermedia API explicitly states that an API follows the principle of Hypermedia As The Engine of Application State (HATEOS) which is a required criteria of being RESTful.

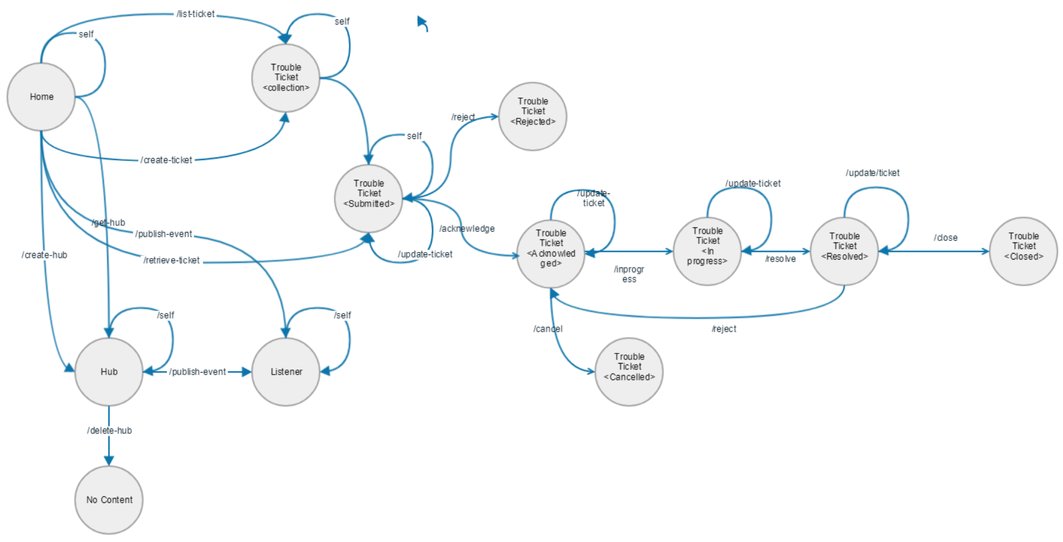
A hypermedia aka web API shall express that an API is naturally following the REST architectural style and hence the web architecture in all aspects. A good summary reflecting on the important role HATEOS is playing for RESTful APIs and an explanation of a RESTful API maturity model, can be found here.

Hypermedia is a way for the server to tell the client what HTTP requests the server, from which the client is free to choose. The server knows what happens, but the client decides what actually happens. The World Wide Web works like this.

Due to the nature of HATEOAS, a web API (and also every web application) can be modelled as a state machine or a workflow.

It describes all states by nodes and the state transitions by vertices from one node to another.

The states usually correspond to a resource, the transitions usually correspond to hypermedia controls, i.e. links that are provided by a resource representation in order to reach another state.



An example is given below:

[_Toc500847761 .anchor]#Figure Trouble Ticket API - State Machine

Chapter 2. Hypermedia Representation Design

Much like web's HTML-based hyperlinks, Web API's response message's body includes links and actions that are available for a given resource, in a given state.

Included along with other fields of a resource representation, links convey the relationships between resources and offer clients a menu of resource-related actions, which are context-sensitive.

A Web API MUST offer clients a consistent way to easily discover the available links within a representation.

2.1. Media Type

The TMF Open API design currently makes use of JSON representations only. The media type is plain JSON ("application/json").

TMF Open APIs offer support for hypermedia links as an extension so without breaking existing clients that rely on the application/json media type.

At the moment the presense of hypermedia extension in the API is not negotiated between client and server based on media type.

The TMF does not use a special media type for hypermedia.

It is assumed that an application supporting hypermedia extension will use the extended representation when the application context is application/json.

2.2. Home Document

The home document serves links to relevant top-level resources of the API and can provide some relevant properties on top.

A home document can:

- Evolve like all the other parts of a hypermedia API. The API provider can introduce new link-relation types "on – the fly". Clients need to be able to cope with getting new link relation types in the home document. By that we mean that they should keep providing the established functionality and they must not crash – obviously they don't need to make use of the additional functionality
- Vary depending in which context it is retrieved. Typically, the available functionality depends on:
  - Which API Consumer has been requesting the home document
  - On which user's behalf is the home document retrieved

In order to be able to do so, the authentication/authorization scheme needs to be aware of both.

The Home URI can be thought of in the same light as a web site home page.

If the API supports hypermedia extension then the HOME URI MUST be supported by the implementation.

A HOME URI only supports an HTTP GET verb.

A HOME URI MUST return a JSON list of linked resource collections as a minimum.

Every TMF Hypermedia Open API MUST have a home document.

An example of Home resource for Trouble Ticket API can be seen below:

```
{
    "_links": {
        "self": {
            "href": "/troubleTicketManagement/v2/home"
        },
        "list-ticket": {
           "href": "/troubleTicketManagement/v2/troubleTicket"
        },
        "retrieve-ticket": {
            "hrefTemplate": "/troubleTicketManagement/v2/troubleTicket/{id}",
            "hrefVars": {
                "id": " /troubleTicketManagement/v2/schema/troubleTicket.json"
            }
        },
        "create-ticket": {
            "title": "Create Ticket",
            "href": "/troubleTicketManagement/v2/troubleTicket",
            "method": "POST",
            "accepts": "application/json",
            "schemaType": "json-schema",
            "schemaUrl": " /troubleTicketManagement/v2/schema/troubleTicket.json"
        },
        "create-hub": {
            "href": "/troubleTicketManagement/v2/hub",
            "method": "POST"
        },
        "get-hub": {
            "hrefTemplate": "/troubleTicketManagement/v2/hub{id}",
            "hrefVars": {
                "href": " /troubleTicketManagement/v2/schema/troubletTicket.json"
            },
            "method": "GET"
```

```
          },
          "delete-hub": {
              "hrefTemplate": "/troubleTicketManagement/v2/hub\{id}",
              "hrefVars": {
                 "href": "/troubleTicketManagement/v2/schema/param/id"
              },
              "method": "DELETE"
          },
          "publish-event": {
          "href": "/troubleTicketManagement/v2/listener",
          "method": "POST"
      }
  }
```

2.3. Linking

A Web API response message body MUST includes links and information about the meaning of the links to allow REST clients to traverse the links to access resources related to the actual resource and perform actions on them.

Links MUST be included, along with fields, within resource state representation.

The link is a JSON object with the name of the object indicating the relationship type, "rel" attribute.

The links MUST be URI's.

Links are represented in TMF Open APIs as JSON objects contained within a _*links* hash that MUST be a direct property of a resource object and have the following structure:

Name	Attribute name	Mandatory?	Description	Type	Remarks
Title	title	No	Its value is a string and is intended for labelling the link with a human-readable identifier	String	

Name	Attribute name	Mandatory?	Description	Type	Remarks
Name	name	No	Its value MAY be used as a secondary key for selecting Link Object which share the same relation type.	String	
URI	href	Conditional	URI	String	Either "href" or "hrefTemplate" to be present
URI Template	hrefTemplate	Conditional	Template URI. Parameters are to be present in flower brackets {} URI Template MUST comply with [RFC6570].	String	Either "href" or "hrefTemplate" to be presenExample: "hrefTemplate": "/widgets/{widget_id}"
URI Variables	hrefVar	Conditional	Variables to be populated in the hrefTemplate	Object	If "hrefTemplate" is present, "hrefVar" should also be present.

Name	Attribute name	Mandatory?	Description	Type	Remarks

URL Variable	<Parameter Name> used in the hrefTemplate	Conditional	Reference to the definition of the variable / parameter	String	Example
					`"hrefTemplate": "/widgets/\{widget_id}",`
					`"hrefVars": \{`
					`"widget_id": "https://example.org/param/widget"`
					`}`

HTTP Method	method	No	HTTP method. Enumerated values GET, POST, PATCH, PUT, DELETE Default: GET	Enum	
Format accepted	accepts	No	Content type format accepted by the URL	String	
Fields		No		Array	
Field name	name	Yes		String	
Field value	value	No			
Field type	type	No			

Name	Attribute name	Mandatory?	Description	Type	Remarks
Request Schema type	schemaType	No	Enumerated values ”json-schema”, “xsd“		
Request schema URL	schemaUrl	No	URL of a schema describing the expected payload for e.g. POST links. Can be used instead of the fields for a more complex structure.		

A *self* link MUST be included in response message body representation.

A *home* link MAY be included in response message body representation.

An example is given below `GET /troubleTicket/42` :

```
{
    "_links": {
        "self": {
            "href": "/troubleTicketManagement/v2/troubleTicket/42"
        },
        "home": {
        "href": "/troubleTicketManagement/v2/home"
    }
}
```

The *self* link for the collection resource SHOULD be exposed as Web Linking HTTP header.

Other special links MAY be included such as pagination links.

An example is given below:

```
{empty}[Response]
Content-Type: application/json
X-Total-Count :50

Link: < https://host:port
/troubleTicketManagement/v2/troubleTicket?count=20&limit=10>;
rel="self",
< https://host:port/ troubleTicketManagement/v2/troubleTicket>;
rel="home",
< https://host:port/
troubleTicketManagement/v2/troubleTicket?count=0&count=10>; rel="first",
< https://host:port
/troubleTicketManagement/v2/troubleTicket?count=30&limit=10>;
rel="next",
< https://host:port
/troubleTicketManagement/v2/troubleTicket?count=10&limit=10>;
rel="prev",
< https://host:port
/troubleTicketManagement/v2/troubleTicket?count=40&limit=10>; rel="last"

[{

    "_links": {
        "self": {
            "href": "/troubleTicketManagement/v2/troubleTicket/6275"
        },
        "home": {
            "href": "/troubleTicketManagement/v2/home"
        }
    },
    "correlationId": "332",
    "creationDate": "2016-10-19T00:00",
    "description": "Compliant over last invoice",
    "href": "/troubleTicketManagement/v2/troubleTicket /6275",
    "id": "6275",
    ...
},
...
]
```

When retrieving the items within a collection resource, *self* link MUST be included for each item.

Other special links MAY be included but it is recommended that other links to be retrieved by the client by going to the self link for each item within a collection.

A resource MAY have multiple links that share the same link relation. This can be represented using an array of links. In this case the attribute name can be used as a secondary key.

An example is given below:

```
{
    "_links": {
        "self": {
            "href": "/troubleTicketManagement/v2/troubleTicket/6275"
        },
        "home": {
            "href": "/troubleTicketManagemet/v2/home"
        },
        "related-party": [{
                "name": "Individual",
                "href": "/partyManagement/v2/individual/123"
            },
            {
                "name": "Organisation",
                "href": "/partyManagement/v2/organisation/345"
            }]
    }
}
```

Here is a full example for Trouble Ticket with all possbile links as per the API state machine defined in chapter 1:

```
{
    "id": "2",
    "type": "Invoice dispute",
    "severity": "low",
    "description": "Invoice number 42 is wrong",
    "status": "Submitted",
    "statusChangeReason": "Initial submission",
    "note": [{
            "author": "Dirk Rejahl",
            "text": "Customer called CC"
        }],
    "_links": {
        "self": {
            "href": "/troubleTicketManagement /v1/troubleTicket/2"
        },
        "acknowledge": {
            "title": "Acknowledge Ticket",
            "href": "/troubleTicketManagement /v1/troubleTicket/2",
            "method": "PATCH",
            "accepts": "application/json",
            "fields": [{
                    "name": "status",
                    "value": "Acknowledged"
                },
                {
                    "name": "statusChangeReason",
                    "type": "string"
                }]
        },
        "reject": {
            "title": "Reject Ticket",
            "href": "/troubleTicketManagement /v1/troubleTicket/2",
            "method": "PATCH",
            "accepts": "application/json",
            "fields": [{
                    "name": "status",
                    "value": "Rejected"
                },
                {
                    "name": "statusChangeReason",
                    "type": "string"
                }]
        }
    }
}
```

2.4. Link Relation Types

Links relations tell clients the meaning of the link allowing REST clients to interact with links.

The IANA provides a registry (http://www.iana.org/assignments/link-relations/link-relations.xhtml) for common link relations.

The table below holds the list of special relationship types supported by the links "rel" property. These are also registered as per the Web Linking standard [RFC5988].

“self”	A link referencing the resource itself.
“related”	A link referencing another resource.

“up”	A link referencing the parent of this resource. Note, there is no “down” relation defined in the Registry Contents.
“first”	A link to the farthest preceding resource in a series of resources. In the context of the pagination this refers to the first page.
“last”	A link to the remotest following resource in a series of resources. In the context of the pagination this refers to the last page.
“next”	A link to the next resource in an ordered series of resources. In the context of the pagination this is the next page.
“previous”	A link to the previous resource in an ordered series of resources. In the context of the pagination this is the previous page.

In adition the TMF Open API defines the following extensions:

Role Specific	An undefined list of relationships, to be taken from the TMF Open API, that represents the relationship type between the target resource and the source resource. e.g For TroubleTicket API we have relationships between a ticket and relatedObject and relatedParty. These relationships can be exposed as specific link relation types: related-party, related-object
home	A link to the API home document
list-{ApiResource}	A link to retrieve all items for API resource collection
retrieve-{ApiResource}	A link to retrieve a single API resource representation
create-{ApiResource}	A link pointing to where an existing API resource representation can be created.

update-{APIResource}	A link pointing to how an existing API resource representation can be updated.
delete–{APIresource}	A link pointing to how an existing API resource reresentation can be deleted.

2.5. Resource Addressing

All the resources MUST be identified by URIs, thus the URL that was used to request the resource is considered its identifier.

In addition to that the identifier can be represented as a Web Linking HTTP header or as a link object – both with the relation type self.

All the resources MUST contain the self link.

Chapter 3. Linked Data

To make the API machine readable, the context of the data passed in the API needs to be communicated to the API client.

This can be done by passing the JSON-LD context to the consumer of the API.

The JSON-LD context SHOULD be passed in the Link HTTP Response header.

An example is given below:

HTTP Response Header

JSON-LD Context Header

```
 Link:<https://host:port/troubleTicketManagemet/v2/tt_context>;
rel="http://www.w3.org/ns/json-ld#context[http://www.w3.org/ns/json-ld#context";
```

HTTP Response Body

*JSON-LD Context – Response Body *

```
{
    "id": "1",
    "href": "/troubleTicketManagemet/v2/troubleTicket/1",
    "correlationId": "TT53482",
    "description": "Customer complaint over last invoice.",
    "severity": "Urgent",
    "@type": troubleTicket",
    "_links": {
        "self": {
            "href": "/troubleTicketManagemet/v2/troubleTicket/1
         }
    }
},
{
    "id": "2",
    "correlationId": "TT53483",
    "description": "Customer asks for information about upgrading products",
    "severity": "Low",
    "@type": "troubleTicket",
    "_links": {
        "self": {
            "href": "/troubleTicketManagemet/v2/troubleTicket/2"
        }
    }
}
```

A sample context for Trouble Ticket API is defined below:

JSON-LD Trouble Ticket Context

```
{
    "@context": {

"troubleTicket": "http://tmf.schema.org/TroubleTicket[http://tmf.schema.org/Troub
leTicket",
        "id": "http://tmf.schema.org/id[http://tmf.schema.org/id",

"correlationId": "http://tmf.schema.org/correlationId[http://tmf.schema.org/corre
lationId",

"description": "http://tmf.schema.org/description[http://tmf.schema.org/descripti
on",
"severity": "http://tmf.schema.org/severity[http://tmf.schema.org/severity",

"creationDate": "http://tmf.schema.org/creationDate[http://tmf.schema.org/creatio
nDate",

"targetResolutionDate": "http://tmf.schema.org/targetResolutionDate[http://tmf.sc
hema.org/targetResolutionDate",
        "status": "http://tmf.schema.org/status[http://tmf.schema.org/status",

"substatus": "http://tmf.schema.org/substatus[http://tmf.schema.org/substatus",

"statusChangeReason": "http://tmf.schema.org/statusChangeReason[http://tmf.schema
.org/statusChangeReason",

"statusChangeDate": "http://tmf.schema.org/statusChangeDate[http://tmf.schema.org
/statusChangeDate",

"resolutionDate": "http://tmf.schema.org/resolutionDate[http://tmf.schema.org/res
olutionDate",

"relatedParty": "http://tmf.schema.org/RelatedParty[http://tmf.schema.org/Related
Party",
        "href": {

"@id": "http://tmf.schema.org/RelatedParty#href[http://tmf.schema.org/RelatedPart
y#href",
            "@type": "@id"
        },

"role": "http://tmf.schema.org/RelatedParty#role[http://tmf.schema.org/RelatedPar
ty#role",

"relatedObject": "http://tmf.schema.org/RelatedObject[http://tmf.schema.org/Relat
edObject",

"involvement": "http://tmf.schema.org/RelatedObject#involement[http://tmf.schema.
org/RelatedObject#involement",
        "reference": {

"@id": "http://tmf.schema.org/RelatedObject#reference[http://tmf.schema.org/Relat
edObject#reference",
            "@type": "@id"
        },
        "note": "http://tmf.schema.org/Note[http://tmf.schema.org/Note",
"date": "http://tmf.schema.org/Note#date[http://tmf.schema.org/Note#date",

"author": "http://tmf.schema.org/Note#author[http://tmf.schema.org/Note#author",
        "text": "http://tmf.schema.org/Note#text",
        "_links": "http://tmf.schema.org/links[http://tmf.schema.org/links",
        "self": "http://tmf.schema.org/self[http://tmf.schema.org/self",
        "@schemaLocation": "http://tmf.schema.org/schemaLocation",
        "@baseType": "http://tmf.schema.org/baseType"
    }
}
```

Note that the context for core TroubleTicket attributes like id, correlationid, description, severity etc. as well as that of keywords like _links, @schemaLocation, @type,@baseType etc. is also defined above. Any extensions to the Trouble Ticket API SHOULD be present in the JSON-LD context.

Also vocabulary for the above context needs to be also defined as a tmf.http://www.schema.org/[schema.org] definition so that the semantics of the attributes are understood.

TMF.schema.org vocabulary provides a specialized vocabulary for the TMF Open APIs and it is built upon the core Schema.org using the extension mechanism provided.

The following provides an example defining the vocabulary of Trouble Ticket entity:

Schema.org Trouble Ticket Vocabulary

Sample content of the vocabulary is provided below.

```
{
    "@context": {
        "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-
ns[http://www.w3.org/1999/02/22-rdf-syntax-ns#",
        "rdfs": "http://www.w3.org/2000/01/rdf-
schema[http://www.w3.org/2000/01/rdf-schema#",

"xsd": "http://www.w3.org/2001/XMLSchema[http://www.w3.org/2001/XMLSchema#"
    },
    "@graph": [{
        "@id": "http://schema.org/TroubleTicket[http://schema.org/TroubleTicket",
        "@type": "rdfs:Class",
        "http://schema.org/isPartOf[http://schema.org/isPartOf": {
            "@id": "http://tmf.schema.org/[http://tmf.schema.org"
        },
        "rdfs:comment": "Trouble Ticket",
        "rdfs:label": "TroubleTicket",
        "rdfs:subClassOf": {
            "@id": "http://schema.org/Thing[http://schema.org/Thing"
        }
    },
    {
        "@id": "http://schema.org/id[http://schema.org/id",
        "@type": "rdf:Property",
        "http://schema.org/domainIncludes[http://schema.org/domainIncludes": [{

"@id": "http://schema.org/TroubleTicket[http://schema.org/TroubleTicket"
            }],
        "http://schema.org/rangeIncludes[http://schema.org/rangeIncludes": {
            "@id": "http://schema.org/Text[http://schema.org/Text"]
        },
        "http://schema.org/isPartOf[http://schema.org/isPartOf": {
                "@id": "http://tmf.schema.org/[http://tmf.schema.org"
        },
        "rdfs:comment": "Ticket ID for a Trouble Ticket entity",
        "rdfs:label": "id"
    },
]}
```

Class **Trouble Ticket (http://schema.org/TroubleTicket)** is defined as a sub class of http://schema.org/Thing and is part of the schema http://tmf.schema.org/.

Class Trouble Ticket

```
{
    "@id": "http://schema.org/TroubleTicket[http://schema.org/TroubleTicket",
    "@type": "rdfs:Class",
    "http://schema.org/isPartOf[http://schema.org/isPartOf": {
        "@id": "http://tmf.schema.org/[http://tmf.schema.org"
    },
    "rdfs:comment": "Trouble Ticket",
    "rdfs:label": "TroubleTicket",
    "rdfs:subClassOf": {
        "@id": "http://schema.org/Thing[http://schema.org/Thing"
    }
}
```

Also attribute **correlation id (http://schema.org/correlationId)** is defined as a Property in the class Trouble Ticket.

Attribute Correlation Id

```
{
    "@id": "http://schema.org/correlationId[http://schema.org/correlationId",
    "@type": "rdf:Property",
    "http://schema.org/domainIncludes[http://schema.org/domainIncludes": [{

"@id": "http://schema.org/TroubleTicket[http://schema.org/TroubleTicket"
        }],
    "http://schema.org/rangeIncludes[http://schema.org/rangeIncludes": {
        "@id": "http://schema.org/Text[http://schema.org/Text"
    },
    "http://schema.org/isPartOf[http://schema.org/isPartOf": {
        "@id": "http://tmf.schema.org/[http://tmf.schema.org"
    },
    "rdfs:comment": "Correlation ID for a Trouble Ticket entity",
    "rdfs:label": "correlationId"
}
```

Similarly enumerated values are also can be defined – Example: Severity - http://schema.org/Severity.

Enumerated -Severity

```
{
    "@id": "http://schema.org/Severity[http://schema.org/Severity",
    "@type": "rdfs:Class",
    "http://schema.org/isPartOf[http://schema.org/isPartOf": {
        "@id": "http://tmf.schema.org/[http://tmf.schema.org"
    },
    "rdfs:comment": "Enumerated Severity values.",
    "rdfs:label": "Severity",
    "rdfs:subClassOf": {
        "@id": "http://schema.org/Enumeration[http://schema.org/Enumeration"
    }
},
{
    "@id": "http://schema.org/Critical[http://schema.org/Critical",
    "@type": "http://schema.org/Severity[http://schema.org/Severity",
    "http://schema.org/isPartOf[http://schema.org/isPartOf": {
        "@id": "http://tmf.schema.org/[http://tmf.schema.org"
    },
    "rdfs:comment": "Critical Severity",
    "rdfs:label": "Critical"
}
```

Other enum values of Severity

Chapter 4. Appendix A: Terms and Abbreviations Used within this Document

4.1. Terminology

Term	Definition	TMF or Outside Source
hypermedia	Hypermedia is data, sent from the server to the client, which explains what client can do next.	
Hypermedia control	A hypermedia controls describes a state transition	
Media type	A media type (also called a content type or MIME type) is a short string identifying the format of the document. Once you know a document's media type you can parse it. You may also be able to understand of its application semantics	

Chapter 5. References

5.1. References

Reference	Description	Source	Brief Use Summary
JSON-LD	**JSON For Linking Data** **Define context to provide clients with additional application semantics of the data**	http://json-ld.org/	Enrich TMF Open API with application semantics in order to enable machine-readable API's.
SCHEMA.ORG	Provides core, basic vocabulary for describing the kind of entities the most common applications need	http://schema.org/	Define vocabulary for TMF Open API Data Model built upon core schema.org vocabulary
Web Linking		https://tools.ietf.org/html/rfc5988	Links in HTTP headers with the Link header field.
Home Documents for HTTP APIs	This document proposes a "home document" format for non-browser HTTP clients.	https://tools.ietf.org/html/draft-nottingham-json-home-02	hrefTemplate and hrefVars are aligned with the draft internet standard.
Richardson Maturity Model		https://martinfowler.com/articles/richardsonMaturityModel.html	

Chapter 6. Administrative Appendix

This Appendix provides additional background material about the TM Forum and this document. In general, sections may be included or omitted as desired; however, a Document History must always be included.

6.1. Document History

6.1.1. Version History

This section records the changes between this and the previous document version as it is edited by the team concerned. Note: this is an incremental number which does not have to match the release number and used for change control purposes only.

Version Number	Date Modified	Modified by:	Description of changes
1.0	25-Aug-2017	Pierre Gauthier TM Forum	Description e.g. first issue of document
2.0.0	08-Nov-2017	Peter Norbury	Update Notice and change document number
2.0.1	12-Dec-2017	Adrienne Walcott	Formatting/style edits prior to publishing
2.0.2	20 Mar 2018	Adrienne Walcott	Updated to reflect TM Forum Approved Status

6.1.2. Release History

This section records the changes between this and the previous Official document release. The release number is the ‘Marketing’ number which this version of the document is first being assigned to.

Release Number	Date Modified	Modified by:	Description of changes
17.5.0	08-Nov-2017	Peter Norbury	first release of document
17.5.1	20 Mar 2018	Adrienne Walcott	Updated to reflect TM Forum Approved Status

6.2. Acknowledgments

This document was prepared by the members of the TM Forum Open API team team:

- Pierre Gauthier, TM Forum, **Editor** and Team Leader
- Nicoleta Stoica, Vodafone, **Author**
- Dirk Rejahl, Infonova, **Contributor**
- Sundar Ramakrishnan, Infosys, **Contributor**
- Amit Sangle, Infosys, **Contributor**

Additional input was provided by the following people:

- Lester Thomas, Vodafone