TM Forum Specification

Trouble Ticket API Conformance Profile

TMF621B

Release 18.0.0

June 2018

Latest Update: TM Forum Release 18.0.0	Member Evaluation
Version 1.2.1	IPR Mode: RAND

NOTICE

Direct inquiries to the TM Forum office:

4 Century Drive, Suite 100
Parsippany, NJ 07054, USA
Tel No. +1 973 944 5100
Fax No. +1 973 944 5110
TM Forum Web Page: www.tmforum.org

TABLE OF CONTENTS

INTRODUCTION - API DESCRIPTION

The Trouble Ticketing API provides a standardized client interface to Trouble Ticket Management Systems for creating, tracking and managing Trouble Tickets among partners as a result of an issue or problem identified by a customer or another system. Examples of Trouble Ticket API originators (clients) include CRM applications, network management or fault management systems, or other Trouble Ticket management systems (e.g. B2B).

The API supports the ability to send requests to create a new ticket specifying the nature and severity of the trouble as well as all necessary related information. The API also includes mechanisms to search for and update existing tickets. Notifications are defined to provide information when a ticket has been updated, including status changes. A basic set of states of a ticket has been specified (as an example) to handle ticket lifecycle management.

Trouble Ticketing API manages Ticket resource:

- A ticket represents a record or an issue raised by requestor that need to be solved, used for reporting and managing the resolution of problems, incidents or request
- Main ticket attributes are its description, severity, type, related dates (creation, expected resolution, resolution), state and related information (change reason and change date), related parties (originator, owner, reviser, etc.), related entities (product, product order, customer bill) and notes

Trouble Ticketing API performs the following operations on ticket

- Retrieval of a ticket or a collection of ticket depending on filter criteria
- Full update of a ticket
- Partial update of a ticket
- Creation of a ticket
- Notification of events on ticket:
    - Ticket state change
    - Ticket change
    - Ticket resoled
    - Ticket created
    - Information required

RESOURCE MODEL CONFORMANCE

API MANDATORY AND OPTIONAL RESOURCES

For the resources defined by the API fill the following table and indicate which ones are mandatory and which ones are optional.

Resource Name	Mandatory or Optional	Comments
TroubleTicket	M	

TICKET MANDATORY AND OPTIONAL ATTRIBUTES

The table below summarizes mandatory and optional attributes for resource "Ticket"

Attribute Name	**Mandatory or Optional**	**Comments**
id	M (in response messages) O (otherwise)	Generated by the server
href	M (in response messages) O (otherwise)	Url for the created resource
externalId	O	
Name	O	
ticketType	M	
creationDate	O	Attribute non patchable
lastUpdated	O	
description	M (for resource creation) O (otherwise)	
reason	O	

Attribute Name	Mandatory or Optional	Comments
severity	M (for resource creation) O (otherwise)	
priority	O	
requestedResolutionDate	O	
expectedResolutionDate	O	
resolutionDate	O	
status	O	
@baseType	O	Populated by the server
@type	O	if not specified will be defaulted to the TMF class name (e.g. TroubleTicket)
@schemaLocation	O	
relatedEntity	O	
statusChange	O	
note	O	
relatedParty	O	
ticketRelationship	O	
channel	O	
attachment	O	

NOTIFICATION MODEL CONFORMANCE

The Pub/Sub models are common and described in the TMF REST Design Guidelines. Use the following templates to describe the Hub Mandatory and Optional attributes and filtering support.

API MANDATORY AND OPTIONAL NOTIFICATIONS

For the Notifications defined by the API the following table indicates which ones are mandatory and which ones are optional.

Notification Name	Mandatory or Optional	Comments
TroubleTicketChangeNotification	M	
TroubleTicketStatusChangeNotification	M	
TroubleTicketCreationNotification	M	
TroubleTicketResolvedNotification	O	
TroubleTicketInformationRequiredNotification	O	

All attributes of the resource associated with the notification are mandatory

API OPERATIONS CONFORMANCE

For every single resource use the following templates and define what operations are optional and what operations are mandatory.

API MANDATORY AND OPTIONAL OPERATIONS

The following table indicates which ones are mandatory and which ones are optional for each one of the resources in the API (default is for all resources).

Uniform API Operation	Mandatory/Optional	Comments
GET	M for all resources	GET must be used to retrieve a representation of a resource
POST	M for resources: TroubleTicket	POST must be used to create a new resource
PATCH	M for resources: TroubleTicket	PATCH must be used to partially update a resource
DELETE	M for resources: TroubleTicket	DELETE must be used to remove a resource. For admin only

API GET OPERATION CONFORMANCE

For every single resource use the following template to specify the mandatory and optional features supported by the GET operation.

Definitions

Filtered Search: A filtered search can be applied using query parameters in order to obtain only the resource entities that meet the criteria defined by the filtering parameters included in the query request. Several elements can be applied to the filtered search. In that case logic, a logical AND is applied to combine the criteria (e.g.?severity=<value> &status=<value>)

Attribute selection (Filtered Response Data): In order to apply a filter and limit the number of attributes included in the response, the GET request can include the "?fields=" query parameter. Several elements can be applied to the filter. In that case, a logical AND is applied to combine the values (e.g.:?fields=severity, status) will provide in the response only the values assigned to attributes category and channel. Attribute selection capabilities are the same for collections retrieval and individual resource queries

All the GET operations in this API share the same status code pattern.

GET	M	
Response Status Code 200	M	
Other Status Codes	NA	

/TROUBLETICKET?FIELDS=...&{FILTERING}

This operation list troubleTicket entities.
Attribute selection is mandatory for all first level attributes.
Filtering is mandatory for first compliance level (L1) and optional otherwise.

/TROUBLETICKET/{ID}?FIELDS=...&{FILTERING}

This operation retrieves a troubleTicket entity.
Attribute selection is mandatory for all first level attributes.
Filtering on sub-resources is optional for all compliance levels.

API POST OPERATION CONFORMANCE

All the POST operations in this API share the same status code pattern.

POST	M	
Status Code 201	M	
Other Status Codes	NA	

/TROUBLETICKET

This operation creates a trouble ticket entity.

Mandatory and Non Mandatory Attributes

The following tables provides the list of mandatory and non mandatory attributes when creating a Ticket, including any possible rule conditions and applicable default values. Notice that it is up to an implementer to add additional mandatory attributes.

Mandatory Attributes	Rule
description	
severity	
ticketType	

Non Mandatory Attributes	Default Value	Rule
externalId		
creationDate		
lastUpdated		
reason		
priority		
requestedResolutionDate		
expectedResolutionDate		
resolutionDate		

Non Mandatory Attributes	Default Value	Rule
status		
@type		
@baseType		
@schemaLocation		
relatedEntity		
statusChange		
note		
relatedParty		
ticketRelationship		
channel		
attachment		

API PUT OPERATION CONFORMANCE

All the PUT operations in this API share the same status code pattern.

PUT	O	
Status Code 201	M	
Other Status Codes	NA	

API PATCH OPERATION CONFORMANCE

All the PATCH operations in this API share the same status code pattern.

PATCH	O	
Status Code 201	M	
Other Status Codes	NA	

/TROUBLETICKET/{ID}

This operation allows partial updates of a ticket entity. Support of json/merge (https://tools.ietf.org/html/rfc7386) is mandatory, support of json/patch (http://tools.ietf.org/html/rfc5789) is optional.

Note: If the update operation yields to the creation of sub-resources or relationships, the same rules concerning mandatory sub-resource attributes and default value settings in the POST operation applies to the PATCH operation. Hence these tables are not repeated here.

Patchable and Non Patchable Attributes

The tables below provide the list of patchable and non patchable attributes, including constraint rules on their usage.

Patchable Attributes	Rule
externalId	
ticketType	
description	
reason	
severity	
priority	
requestedResolutionDate	
expectedResolutionDate	
resolutionDate	
status	
relatedEntity	
note	
relatedParty	
ticketRelationship	
channel	
attachment	

Non Patchable Attributes	Rule
id	
href	

Non Patchable Attributes	Rule
creationDate	
lastUpdated	
statusChange	
@baseType	
@type	
@schemaLocation	

API DELETE OPERATION CONFORMANCE

All the DELETE operations in this API share the same status code pattern.

DELETE	M	Used by admin only
Status Code 200	M	
Other Status Codes	NA	

/TROUBLETICKET/{ID}

This operation deletes a trouble ticket entity.

API CONFORMANCE TEST SCENARIOS

This section describes the test scenarios required for the basic CONNECT certification of the API.

Test Cases must be executed in the order defined for each resource because the result from one of the scenarios will be input for the next one.

Requests must be addressed to the endpoint provided for certification, specifically they must be addressed to the URI defined by the concatenation of the {apiRoot} and the specific resource, where the {apiRoot} is defined as **{serverRoot}/troubleTicket/v1**, where {serverRoot} defines the certification endpoint

RESOURCEXYZ RESOURCE TEST CASES

Nominal Scenarios

TC_Trou_N1 – Create new trouble ticket with minimum required information

- Send a POST message to {apiRoot}/troubleTicket/ with the following contents in the BODY

```
{
   "description": "<anytext>",
   "severity": "High",
   "ticketType": "device"
}
```

- Wait for a response from the server with the following characteristics
  - Response Code 201-Created
  - Include a location header in the body set to /{apiRoot}/troubleTicket/{IDtt1} where {IDtt1} indicates the identifier assigned by the server to the new ticket resource
  - The response message includes all mandatory parameters (including description, severity and type that were not sent in the original request)
  - The body of the response matches the values set in the original request
- Send a GET message to /{apiRoot}/troubleTicket/

- Wait for a response from the server with the following characteristics
    - Response Code 200-OK
    - The body of the response includes one TroubleTicket resource with ID set to {IDtt1}, the same identifier as assigned by the server to the new resource created
    - The response message includes all mandatory parameters
    - The body of the response for the resource with identifier {IDtt1} matches the values set in the original request
- Send a GET message to /{apiRoot}/troubleTicket/{IDtt1}
- Wait for a response from the server with the following characteristics
    - Response Code 200-OK
    - The response message includes all mandatory parameters
    - The body of the response includes a TroubleTicket resource structure that matches the values in the original request

TC_Trou_N2 – Create new trouble ticket with minimum set of parameters supported by server

- Send a POST message to {apiRoot}/troubleTicket/ with the following contents in the BODY

```
{
  "description": "nanana",
  "severity": "Low",
  "ticketType": "connectivity",
  "status": "nanana",
  "externalId": "123",
  "note":
  [
    {
      “author”: “writer n2_1”,
      “text”: “This is the first note in N2”
    },
    {
```

```
            "author": "writer n2_2",
            "text": "This is the second note in N2"
        }
    ]
}
```


- Wait for a response from the server with the following characteristics
    - Response Code 201-Created
    - Include a location header in the body set to /{apiRoot}/troubleTicket/{IDtt2} where {IDtt2} indicates the identifier assigned by the server to the new TroubleTicket resource
    - The response message includes all mandatory parameters (including creationDate, status and statusChangeDate that were not sent in the original request)
    - The body of the response matches the values set in the original request
- Send a GET message to /{apiRoot}/troubleTicket/
- Wait for a response from the server with the following characteristics
    - Response Code 200-OK
    - The body of the response includes one TroubleTicket resource with ID set to {IDtt2}, the same identifier as assigned by the server to the new resource created
    - The response message includes all mandatory parameters
    - The body of the response for the resource with identifier {IDtt2} matches the values set in the original request
- Send a GET message to /{apiRoot}/troubleTicket/{IDtt2}
- Wait for a response from the server with the following characteristics
    - Response Code 200-OK

- The response message includes all mandatory parameters
- The body of the response includes a TroubleTicket resource structure that matches the values in the original request

TC_Trou_N3 – Search for trouble tickets with specific characteristics

- Send a GET message to /{apiRoot}/troubleTicket
- Wait for a response from the server with the following characteristics
  - Response Code 200-OK
  - The body of the response includes at least two ticket resources referring to {IDtt1} and {IDtt2}
  - The body of the response for the resource with each identifier matches the values in the corresponding original request
- Send a GET message to /{apiRoot}/ troubleTicket?severity=High
- Wait for a response from the server with the following characteristics
  - Response Code 200-OK
  - The body of the response includes one TroubleTicket resource referring to {IDtt1} and there is no reference to TroubleTicket resource {IDtt2}
  - The response message includes all mandatory parameters
  - The body of the response for the resource with identifier {IDtt1} matches the values in the original request
- Send a GET message to /{apiRoot}/troubleTicket?type=connectivity
- Wait for a response from the server with the following characteristics
  - Response Code 200-OK

- The body of the response includes one TroubleTicket resource referring to {IDtt2} and there is no reference to TroubleTicket resource {IDtt1}

- The response message includes all mandatory parameters

- The body of the response for the resource with identifier {IDtt2} matches the values in the original request

TC_Trou_N4 – Filtered retrieval of Tickets

- Send a GET message to /{apiRoot}/troubleTicket/{IDtt1}?fields=description

- Wait for a response from the server with the following characteristics

  - Response Code 200-OK

  - The body of the response includes one TroubleTicket resource referring to {IDtt1} and including only attributes name and status, matching the values in the original request

- Send a GET message to /{apiRoot}/ troubleTicket /{IDtt2}?fields=severity,status

- Wait for a response from the server with the following characteristics

  - Response Code 200-OK

  - The body of the response includes one TroubleTicket resource referring to {IDtt2} and including only attributes severity and status, matching the values in the original request

  Notice that this test case is using parameters ”description”, ”severity” and ”status” to filter the data included in the response but any other parameter could be used

TC_Trou_N5 – Filtered Search and Filtered data response

- Send a GET message to /{apiRoot}/troubleTicket?severity=High&fields=description

- Wait for a response from the server with the following characteristics

  - Response Code 200-OK

  - The body of the response includes one TroubleTicketresource referring to {IDtt1} and there is no reference to TroubleTicket resource {IDtt2}

  - The body of the response for the resource with each identifier includes only attribute description, matching the values in the corresponding original request

  Notice that this test case is using the parameter ”description” to filter the data included in the response but any other parameter could be used

Error Scenarios

TC_Trou_E1 – Unknown Trouble Ticket identifier

- Send a GET message to /{apiRoot}/troubleTicket/{IDtt3}, where {IDtt3} does not match any of the identifiers previously creted in the server

- Wait for a response from the server with the following characteristics

  - Response Code 404-Not Found

TC_Trou_E2 – Invalid Request – Missing mandatory parameter

- Send a POST message to {apiRoot}/troubleTicket/ with the following contents in the BODY.

```
{
  “description”: “<anytext>”,
  “severity”: “High”
}
```

Notice that this request is missing mandatory parameter ”type” but any other mandatory parameter could be used

- Wait for an error response from the server indicating the mandatory parameter is missing in the request

TC_Trou_E3 – Invalid Request – Missing parameter mandatory in context

- Send a POST message to {apiRoot}/troubleTicket/ with the following contents in the BODY.

```
{
  “description”: “<anytext>”,
  “severity”: “High”,
  “ticketType”: “problem”,
  “note”:
    {
      “author”: “writer e3_1”
    }
}
```

Notice that this request is missing mandatory parameters “text” when information element “note” is included in the request, but any other parameter that becomes mandatory based on the context could be used

Wait for an error response from the server indicating the mandatory parameter is missing in the request.

ACKNOWLEDGEMENTS

VERSION HISTORY

Version Number	Date	Release led by:	Description
1.0	07/15/2015	Pierre Gauthier TM Forum pgauthier@tmforum.org	First Release of Draft Version of the Document.
1.1	03/15/2017		Updated version including Test scenarios
1.2	4/Apr/2018	Jacob Avraham jacoba@amdocs.com	Align with REST Design Guideline (DG3). Enhanced model with new requirements
1.2.1	06-Jun-2018	Adrienne Walcott	Formatting/style edits prior to R18 publishing

RELEASE HISTORY

Release Number	Date	Release led by:	Description
Release 18.0.0	06-Jun-2018	Pierre Gauthier Jacob Avraham	Initial Release.