Frameworx Specification

Trouble Ticket API Conformance Profile

TMF661
Release 16.5.0
December 2016

Latest Update: Frameworx Release 16.5	Member Evaluation
Version 1.0.0	IPR Mode: RAND

NOTICE

Direct inquiries to the TM Forum office:

240 Headquarters Plaza,
East Tower – 10th Floor,
Morristown, NJ 07960 USA
Tel No. +1 973 944 5100
Fax No. +1 973 944 5110
TM Forum Web Page: www.tmforum.org

TABLE OF CONTENTS

NOTICE....2
Table of Contents....3
List of Tables....4
Introduction....5
API DESCRIPTION....6
RESOURCE MODEL CONFORMANCE....7
Trouble Ticket API MANDATORY AND OPTIONAL RESOURCES....7
Ticket resource MANDATORY AND OPTIONAL ATTRIBUTES....7
API OPERATIONS CONFORMANCE....10
Trouble Ticket MANDATORY AND OPTIONAL OPERATIONS....10
API GET FILTERING OPERATION CONFORMANCE....11
Filtering in Ticketing resource....11
GET /troubleTicket /ticket/....12
GET /troubleTicket /ticket /{ID}....12
API POST OPERATION CONFORMANCE....13
POST to /troubleTicket /ticket/....13
API PATCH OPERATION CONFORMANCE....16
API DELETE OPERATION CONFORMANCE....17
API CONFORMANCE TEST SCENARIOS....18
Trouble Ticket resource TEST CASES....18
Acknowledgements....25
Release History....25

LIST OF TABLES

N/A

INTRODUCTION

The following document is the REST API Conformance for the Trouble Ticket API (TMF621).

API DESCRIPTION

The Trouble Ticketing API provides the 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).

A trouble ticket represents a record used for reporting and managing the resolution of resource problems.

RESOURCE MODEL CONFORMANCE

Trouble Ticket API MANDATORY AND OPTIONAL RESOURCES

Resource Name	Mandatory or Optional	Comments
Ticket	M	

Ticket resource MANDATORY AND OPTIONAL ATTRIBUTES

Attribute Name	Mandatory or Optional	Comments
id	M (in response messages) O (otherwise)	Generated by the server and provided in the response upon resource creation. Accepted in entity-creation requests if the server supports the incoming identifier as the reference to create new resources
href	M (in response messages) O (otherwise)	Value in response must be the same as the one set in Location header provided upon entity creation
correlationId	O	
description	M	
severity	M	This is the severity as identified by the requestor (perceived severity). The server may change its

		value after processing the request.
type	M	
creationDate	M/O	Refers to the date when the ticket entity was created in the server. Only required in response messages from the server when providing the Ticket entity contents (responses to create-POST and read-GET the entity)
targetResolutionDate	O	
status	M/O	Only required in response messages from the server when providing the Ticket entity contents (responses to create-POST and read-GET the entity)
subStatus	O	
statusChangeReason	O	
statusChangeDate	O	
resolutionDate	O	
relatedParty	O	Array of structures
href	M (if relatedObject included) O (otherwise)	
role	O	

	name	O	
	validFor	O	
relatedObject		O	Array of structures
	reference	M (if relatedObject included) O (otherwise)	
	involvement	O	
note		O	Array of structures
	date	M (in response messages and if note included) O (otherwise)	Only required in response messages from the server
	author	M (if note included) O (otherwise)	
	text	M (if note included) O (otherwise)	

API OPERATIONS CONFORMANCE

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

Trouble Ticket MANDATORY AND OPTIONAL OPERATIONS

Uniform API Operation	Mandatory/Optional	Comments
GET	M	GET must be used to retrieve a representation of a resource
POST	M	POST must be used to create a new resource
PUT	O	PUT must be used to completely update a resource identified by its resource URI
PATCH (JSON-MERGE)	O	PATCH must be used to partially update a resource
DELETE	O	DELETE must be used to remove a resource

API GET FILTERING OPERATION CONFORMANCE

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>)

Filtered Data (Attribute selection): 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

Filtering in Ticketing resource

Attribute name	**Filtered search First Level**	**Filtered search N Level**	**Attribute Selection First Level**	**Attribute Selection N Level**
id	NA	NA	M	NA
href	NA	NA	M	NA
correlationId	O	NA	M	NA
description	O	NA	M	NA
severity	M	NA	M	NA
type	M	NA	M	NA
creationDate	O	NA	M	NA
targetResolutionDate	O	NA	M	NA
status	O	NA	M	NA

subStatus	O	NA	M	NA
statusChangeReason	O	NA	M	NA
statusChangeDate	O	NA	M	NA
resolutionDate	O	NA	M	NA
relatedParty	NA	O	M	O
relatedObject	NA	O	M	O
note	NA	O	M	O

GET /troubleTicket /ticket/

Filtered Search: A filtered search can be applied using the following filtering criteria

- status: To obtain the list of tickets that are in a given status
- severity: To obtain the the list of tickets that are stored in server with a given severity
- other optional attributes as defined in the table above

Filtered Data: A filtered response can be requested for the following attributes using the "?fields=" query parameter

- Any of the attributes in the first level of Ticket resource definition

GET /troubleTicket /ticket /{ID}

Filtered Search: A filtered response can be requested for the following attributes using the "?fields=" query parameter

- Any of the attributes in thefirst level of Cutsomer resource definition

API POST OPERATION CONFORMANCE

POST to /troubleTicket /ticket/

This Uniform Contract operation is used to create a Ticket resource in the server.

The response to this operation must include a Location header set to /troubleTicket /ticket /{ID} where {ID} indicates the identifier assigned by the server to the new Product Offering resource created

POST	M	
Response Status Code 201	M	
Other Status Codes	NA	

The following table indicates attributes that are required to be sent when creating a new Ticket resource as well as attributes with special considerations. All other attributes defining the resource are not required to be sent as aprt of the BODY of the POST request message:

Attribute name	Mandatory	Default	Rule
id	N		Accepted in entity-creation requests if the server supports the incoming identifier as the reference to create new resources
description	Y		
severity	Y		This is the severity as identified by the requestor (perceived severity). The server may change its value after processing the request.
type	Y		
relatedParty.href	N		Mandatory if relatedParty included

			The consumer must indicate the identifier for every relatedParty assigned to the ticket
relatedObject.reference	N		Mandatory if relatedObject included The consumer must indicate the identifier for every relatedObject assigned to the ticket
note.author			Mandatory for each note entity included
note.text			Mandatory for each note entity included

The response from the server must include a BODY with the contents of the new resource created, filled with at least the same information elements that were included in the request and are supported by the server. Notice that the value stored by the server may be different than the one set in the request (e.g.: severity may be differently understood by the server after processing than the one perceived by the requestor)

If the POST request includes optional parameters (as per the model resource definition) that are not supported by the server, then the server must reject the request (replying with a 4xx error response) indicating the parameter not supported.

The following parameters must be supported by the server when included in the request to create a new resource

- description
- severity
- type
- status
- correlationId
- note
- note.author
- note.text

The BODY of the response from the server must include attribute “href” set to the same value as the one in the Location header.

The server must include in the BODY of the response, even if they are not included in the request, the following attributes that are mandatory in the definition of a Ticket as per the resource model defined

- id
- href
- status
- creationDate

The BODY of the response from the server must include attribute “href” (or “reference”) under each one of the entities within the Ticket model that can be addressed individually and were included in the response. This applies to any of the following entities

- relatedParty
- relatedObject

API PATCH OPERATION CONFORMANCE

This section defines which attributes are patchable.

Since PATCH operation is optional and not included in the basic CONNECT certification this is not applicable in this conformance document.

API DELETE OPERATION CONFORMANCE

This section defines what operations can be used to delete a Ticket resource.

Since DELETE operation is optional and not included in the basic CONNECT certification this is not applicable in this conformance document

API CONFORMANCE TEST SCENARIOS

This section describes the test scenarios required for the basic CONNECT certification of Trouble Ticket 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**, being {serverRoot} defines the certification endpoint

Trouble Ticket resource TEST CASES

Nominal Scenarios

TC_Trou_N1 – Create new Ticket with minimum required information

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

```
{
    "description": "<anytext>",
    "severity": "High",
    "type": "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 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",
    "type": "connectivity",
    "status": "nanana",
    "correlationId": "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 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",
    "type": "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

RELEASE HISTORY

Release Number	Date	Release led by:	Description
Release 1.0.0	12/01/2016	Pierre Gauthier TM Forum pgauthier@tmforum.org	Version issued for Fx16.5