TM Forum Specification

User Roles and Permissions API REST Specification

TMF672
Release 17.0.1
November 2017

Latest Update: TM Forum Release 17	TM Forum Approved
Version 0.3.1	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

LIST OF TABLES

N/A

INTRODUCTION

The following document is the specification of the REST API for User Roles & Permissions. It includes the model definition as well as all available operations for creating user permissions to access manageable assets.

For the purpose of this API, the following definitions apply

- A **manageable asset** is the realization of something that can be used and managed by users (e.g.: any of the resources created as part of a purchased product, a service provided to individuals, a block of personal data of an individual, a shopping cart entity…..).
- An **user** is the individual who can make use and manage the functions exposed by a given manageable asset. It can be the existing registered customer or any other individual who has been granted access to use and/or manage the asset
- A basic **permission** provides information regarding the access privileges of a given user over manageable assets (or different functions within each asset).
- A **privilege** defines an independent allowed access level over any of the operations that can be performed over a given asset (e.g.: CRUD on the different menu/function/UI elements)
- A **user role** is defined as the entity that defines a set of privileges covering various functions and/or manageable assets. When a user is assigned a given role then it is actually allocated all the privileges defined for that roletype and the corresponding permissions are created for that user

This API allows the following operations

- Create new permission granted by an individual to another individual to access his owned manageable assets
- Read existing permissions. It can be filtered for specific criteria (e.g.: date recorded, granter, …)
- Read specific existing permission
- Modify specific existing permission (total or partial update)
- Read permissions recorded for an specific user as granter
- Read permissions recorded for an specific user as grantee
- Read permissions recorded for an specific asset
- Create new user role
- Read existing user roles
- Read specific existing user role
- Assigning specific user role to an individual (Party) over a given manageable asset.

Consuming this API must be done following a secured mechanism (e.g.: OAuth2.0) so that permissions to access manageable assets is only granted by consumers holding a valid authorization to operate on those manageable assets and grant permissions.

This API assumes that once a product is purchased by a customer (under a given account), as part of the product instantiation during the provisioning process, if a manageable asset is created under that product instance (e.g.: an eCare system registration, an account into a digital service platform, …) then this manageable assets will be assigned as owner to the individual that has admin rights over the customer and account entity under which the product was purchased. This association will be the first permission registered in the system (root permission) over the specific manageable asset, granting that individual (or another one if the purchasing process allows defining another admin for the manageable assets created) owner access to that asset and then the owner can use this API to grant access, with different access levels, to other individuals (users).

SAMPLE USE CASES

This section includes a set of main use cases that can be performed with this API. Additional use cases can be generated using the operations and resources defined in this specification.

Use Case 1: New permission created

Description

The main purpose of this use case is the creation of a new permission by an individual so that another individual is authorized to get access to some of his assets. For instance a user that owns a TV service can grant access to another user in order to make use of some of the functions within the service, for instance view only children movies, configure TV service or view documentaries.

Main Actors

- The owner of the assets (granter)
- The receiver of the permission (user)

  Prerequisite: This API assumes that once a product is purchased by a customer (under a given account), as part of the product instantiation during the provisioning process, if a manageable asset is created under that product instance (e.g.: an eCare system registration, an account into a digital service platform, …) then this manageable assets will be assigned as owner to the individual that has admin rights over the customer and account entity under which the product was purchased.

Use Case Steps

i. The owner of the assets, sends a request to allocate a permission to another user on his assets

ii. The Operator receives the permission creation request including the following minimum information
    a. The period during which the permission is valid
    b. Impacted user that is being granted access
    c. Information on the manageable assets the user is granted access
    d. The level of access granted over each of the manageable assets (indicating the functions that can be accessed on the asset and the actions that can be performed on those functions)

iii. The Operator confirms that the requestor is authorized to assign permissions on the referenced manageable assets (i.e.: either is the owner or has access to the assets with appropriate level). This could be based on just the requestor identifier or via a more sophisticated token-based authorization mechanisms

iv. The Operator allocates the requested access level for the referred manageable assets to the individual that has been granted by the owner.

Example of API Usage in the Context of the Use Case

The following API interactions support the use case:

- The owner of the manageable assets consumes the service offered by the Operator to create a new permission record.

Success Outcome

After completion of these API interactions, the individual that has been granted access to the referred manageable assets can make use according to the access level granted.

Use Case 2: New User Role assigned to an individual

Description

The main purpose of this use case is the creation of a new user role indicating the access level authorized on a given set of manageable assets to whoever is allocated this role.

Main Actors

- The admin operator that generates user roles
- The owner of the assets (granter)
- The individual allocated a given role (user)

  Prerequisite: This API assumes that once a product is purchased by a customer (under a given account), as part of the product instantiation during the provisioning process, if a manageable asset is created under that product instance (e.g.: an eCare system registration, an account into a digital service platform, …) then this manageable assets will be assigned as owner to the individual that has admin rights over the customer and account entity under which the product was purchased.

Use Case Steps

i. The admin operator, sends a request to create a new user role including the following minimum information
   a. The level of access granted over a set of functions that can be accessed on an asset and the actions that can be performed on those functions

ii. Once the user role is defined, the owner of the assets (or the admin), sends a request to allocate a permission to another user based on the user role definition, including the following minimum information
   a. The period during which the permission is valid
   b. Impacted user that is being granted access
   c. Information on the manageable assets the user is granted access
   d. The user role allocated to the user on teh referenced asset

iii. The Operator confirms that the requestor is authorized to assign permissions on the referenced manageable assets (i.e.: either is the owner or has access to the assets with appropriate level). This could be based on just the requestor identifier or via a more sophisticated token-based authorization mechanisms

iv. The Operator allocates the requested access level for the referred manageable assets to the individual that has been granted by the owner.

Example of API Usage in the Context of the Use Case

The following API interactions support the use case:

- The admin operator consumes a service to create a new user role
- The owner of the manageable assets consumes the service offered by the Operator to create a new permission record based on assigning a role to a user over a given asset.

Success Outcome

After completion of these API interactions, the individual that has been granted access to the referred manageable assets can make use according to the access level defined for the allocated user role.

RESOURCE MODEL

PERMISSION RESOURCE

The Permission resource represents the entitlement given by an individual (granter) to another individual (user) to get access to a set of his owned manageable assets. One single permission resource can hold information referring to privileges granted for multiple manageable assets.

A user permission is a specialization of a permission holding only information regarding privileges given to a specific user.

An asset permission is a specialization of a permission holding only information regarding privileges given over a specific asset.

Both UserPermission and AssetPermission inherit all attributes of a Permission.

The following is an example of the data structure of a Permission resource.

```
{
  "id":"Prms123",
  "href":"endpoint/usersandroles/v1/permission/Prms123",
  "date": "2016-03-25T12:00:00",
  "description": "this is the permission to access TV and mobile line",
  "period":
    {
     "startDateTime":"2016-03-25T12:00:00",
     "endDateTime": "2017-03-25T12:00:00"
    },
  "user":
   {
     "id"="u123",
     "href": "http://server:port/PartyManagement/individual/u123",
     "name": "John Doe"
   },
  "granter":
   {
     "id"="u987",
     "href": "http://server:port/PartyManagement/individual/u987",
     "name": "Jim Grants"
   },
  "privilege":
   [
    {
     "manageableAsset":
      {
       "id"= "Asset987",
       "entityType"="IPTV license"
      },
     "function":"Netflix configuration",
     "action":"R&W"
    },
    {
     "manageableAsset":
      {
       "id"= "Asset987",
       "entityType"="IPTV license"
      },
     "function":"Sport basic package",
     "action":"watch"
    },
    {
     "manageableAsset":
      {
       "id"="Asset123",
       "entityType"="mobile line"
      },
     "function":"last calls",
     "action":"R/O"
    }
   ]
}
```

FIELD DESCRIPTIONS

Element	Type	Mandatory in API messages	Description
id	String	Yes in response	Unique Identifier within the server for the permission.
href	anyURI	Yes in response	A resource URI pointing to the resource in the server that stores the detailed information. This is typically the resource url to retrieve individual details for the specific permission
date	dateTime	Yes in response	Date when the permission was created
description	String	No	Text describing the permission
period	TimePeriod	Yes in request and response	Date Interval during which the permission is in effect
user	InvolvementIdentification Ref	Yes in request and response	A reference to the individual that is given the permission to access different assets/operations
granter	InvolvementIdentification Ref	Yes in response	A reference to the individual that is granting permission to access his owned assets/operations

Element	Type	Mandatory in API messages	Description
privilege	Array of Privilege entity structure (Business interaction item)	Yes in request and response if not assetUserRoles included (at least one entry in the array)	Array including information about the individual entitlements to access specific assets to perform specific operations
assetUserRole	Array of AssetInvolvementRole entity structure	Yes in request and response if not privileges included (at least one entry in the array)	Array including information about the preconfigured set of entitlements to access specific assets to perform specific operations

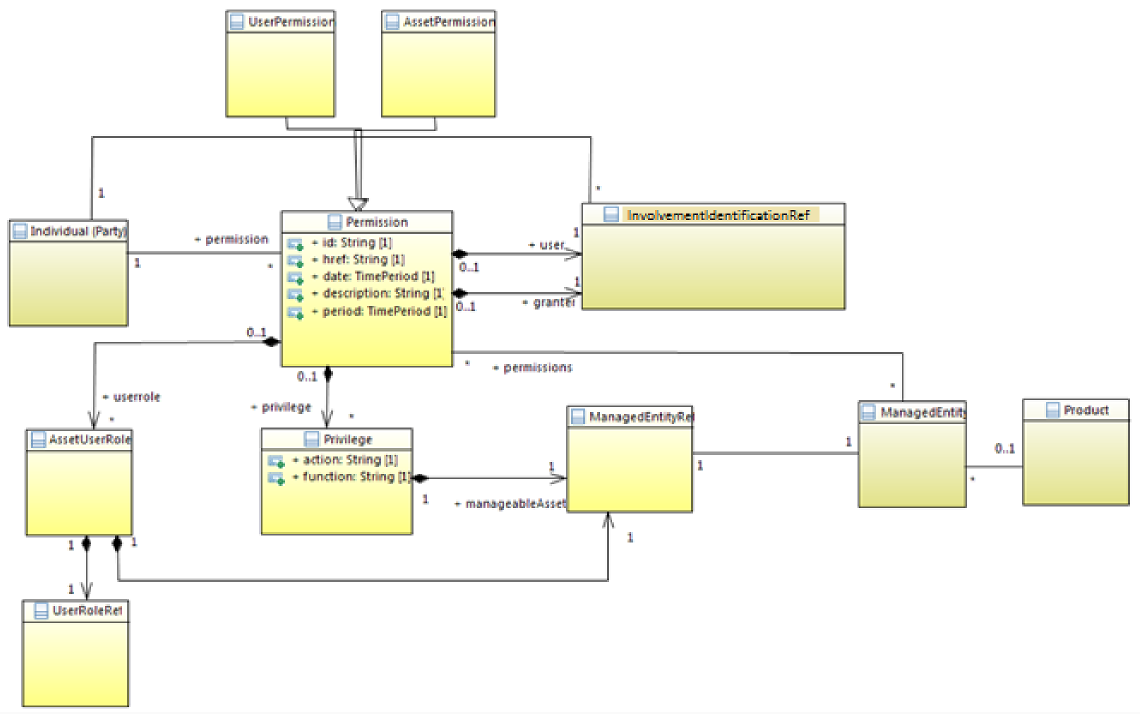

Figure 1 Permission resource model

The following table shows how the entities in the model align with TM Forum SID entities.

Entity in this API	Entity in SID

user	InvolvementIdentification
granter	InvolvementIdentification
permission	Agreement (BusinessInteraction)
privilege	AgreementItem (BusinessInteractionItem)
manageableAsset	ManagedEntity
assetUserRole	InvolvementRole

The definition of user/granter as InvolvementIdentification (which is the base class for PartyUser and ResourceUser is intended to support future scenarios where the entity managing assets is not an individual but a machine

Field Descriptions

Privilege: Detailed information of individual access entitlement

Field	Type	Mandatory in API msg	Description
manageableAsset	ManagedEntityRef	Yes	Reference to the manageable asset whose access is being granted by the owner to another user
function	String	No	Specific function that can be managed over a given asset. Only required if an specific operation over the asset needs to be identified
action	String	Yes in request and response	Level of access granted as part of the permission

ManagedEntityRef: Reference to a managed asset

Field	Type	Mandatory in API msg	Description
id	String	Yes in request and response	Unique identifier of referenced entity
href	String	No	Reference to a resource in the server including details on the referenced entity
entityType	String	Yes in request and response	Type of asset (e.g.: mobile line, video platform license, …)

InvolvementIdentificationRef: link to the resource that holds information about another entity (user) with relationship to the permission. An individual (party) could have multiple users associated (as employee, as customer, …)

Field	Type	Mandatory in API msg	Description

id	String	Yes in request and response	Unique identifier for the partyUser entity
href	String	Yes in response	A resource URI pointing to the resource in the OB that stores the party entity information
name	String	No	Name of the partyUser

AssetInvolvementRole: Detailed information of individual user role

Field	Type	Mandatory in API msg	Description
manageableAsset	ManagedEntityRef	Yes in request and response	Reference to the manageable asset whose access is being granted by the owner to another user
userRole	UserRoleRef	Yes in request and response	Specific preconfigured set of entitlements

UserRoleRef: link to the resource that holds information about another entity (userRole)

Field	Type	Mandatory in API msg	Description
id	String	Yes in request and response	Unique identifier for the user role entity
href	String	Yes in response	A resource URI pointing to the resource in the OB that stores the user role entity information
role	String	No	Involvement role used to define the user role

USER ROLE RESOURCE

The UserRole resource represents the part played by an individual in relation to being granted to access a set of manageable assets.

The following is an example of the data structure of a User Role resource.

```
{
  "id":"UR123",
  "href":"endpoint/usersandroles/v1/role/UR123",
  "involvementRole":"configure IPTV and watch news",
  "entitlement":
   [
     "function":"Netflix configuration",
     "action":"R&W"
    },
    {
     "function":"Sport basic package",
     "action":"watch"
    }
   ]
}
```

FIELD DESCRIPTIONS

Element	Type	Mandatory in API messages	Description
id	String	Yes in response	Unique Identifier within the server for the user role.
href	anyURI	Yes in response	A resource URI pointing to the resource in the server that stores the detailed information. This is typically the resource url to retrieve individual details for the specific user role
involvementRole	String	Yes in request and response	Indication of the part that a user plays in its involvement with a manageable asset (product, service or resource)
entitlement	Array of entilement entity structure (Business interaction item)	Yes in request and response (at least one entry in the array)	Array including information about individual entitlements to define access levels to operate over a given function that can be included in an asset

Entitlement: Detailed information of individual access entitlement

Field	Type	Mandatory in API msg	Description
function	String	No	Specific function that can be managed over a given asset. Only required if an specific operation over the asset needs to be identified
action	String	Yes in request and response	Level of access granted as part of the user role

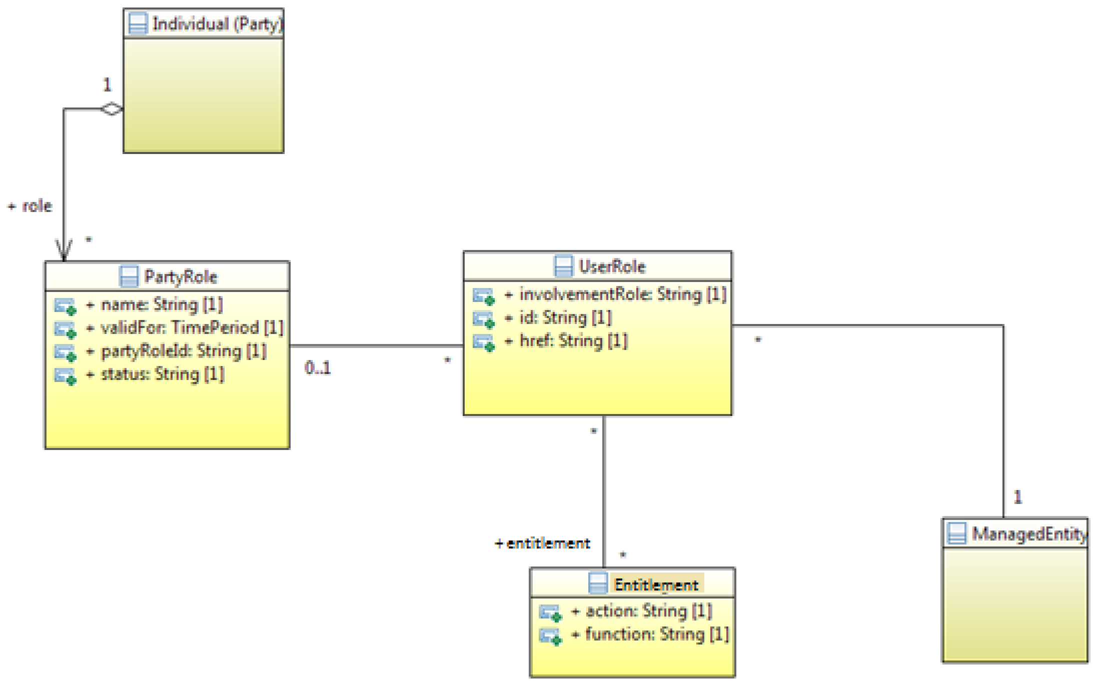

Figure 2 User Role resource model

The following table shows how the entities in the model align with TM Forum SID entities.

Entity in this API	Entity in SID
userRole	InvolvementRole
individual	Party
entitlement	AgreementItem (BusinessInteractionItem)
manageableAsset	ManagedEntity

Notification Resources Models

The following notifications are defined for this API

Notifications related to Permission:

- PermissionCreationNotification
- PermissionChangeNotification

Notifications related to User Role:

- UserRoleCreationNotification
- UserRoleChangeNotification

Permission Creation Notification

Notification sent when a new permission has been created.

Json representation sample

We provide below the json representation of an example of a 'PermissionCreationNotification ' notification object.

```
{
   "eventId":"00001",
   "eventTime":"2016-11-16T16:42:25-04:00",
   "eventType":"PermissionCreationNotification",
    "event": {
      "productOrder" :
         {-- SEE Permission RESOURCE SAMPLE --}
   }
}
```

Permission Change Notification

Notification sent when the definition of a previous permission has been modified.

Json representation sample

We provide below the json representation of an example of a 'PermissionChangeNotification' notification object.

```
{
   "eventId":"00001",
   "eventTime":"2016-11-16T16:42:25-04:00",
   "eventType":"PermissionChangeNotification",
    "event": {
      "productOrder" :
         {-- SEE Permission RESOURCE SAMPLE --}
   }
}
```

UserRole Creation Notification

Notification sent when a new user role has been created.

Json representation sample

We provide below the json representation of an example of a 'UserRoleCreationNotification' notification object.

```
{
   "eventId":"00001",
   "eventTime":"2016-11-16T16:42:25-04:00",
   "eventType":"UserRoleCreationNotification",
    "event": {
      "productOrder" :
         {-- SEE Permission RESOURCE SAMPLE --}
   }
}
```

Permission Change Notification

Notification sent when the definition of a previous user role has been modified.

Json representation sample

We provide below the json representation of an example of a 'UserRoleChangeNotification' notification object.

```
{
   "eventId":"00001",
   "eventTime":"2016-11-16T16:42:25-04:00",
   "eventType":"UserRoleChangeNotification",
    "event": {
      "productOrder" :
         {-- SEE Permission RESOURCE SAMPLE --}
   }
}
```

API OPERATION TEMPLATES

For every single of operation on the entities use the following templates and provide sample REST requests and responses.

Remember that the following Uniform Contract rules must be used:

Operation on Entities	Uniform API Operation	Description
Query Entities	GET Resource	GET must be used to retrieve a representation of a resource.
Create Entity	POST Resource	POST must be used to create a new resource
Partial Update of an Entity (Optional)	PATCH Resource	PATCH must be used to partially update a resource
Complete Update of an Entity (Optional)	PUT Resource	PUT must be used to completely update a resource identified by its resource URI
Remove an Entity (Optional)	DELETE Resource	DELETE must be used to remove a resource

Filtering and attribute selection rules are described in the TMF REST Design Guidelines.

PERMISSIONS RESOURCE

GET /usersandroles/v1/permission

Description:

The Application invokes this operation to retrieve the permissions stored in the server.

The request could include a filter to retrieve only the permissions that match specific criteria (e.g.: created within a given date range).

Notice that this operation is expected to include some query parameter since it may not be feasible to retrieve all the permissions in the system. At least one filter is expected in order to prevent from having a response to an unbounded collection but each specific server implementation must define the limits on the maximum number of elements in the response.

Behavior:

Status Code	Description
200	Permissions information was returned successfully
400	Request Error
500	The server encountered an unexpected condition which prevented it from fulfilling the request
Other	The server may use other HTTP error status codes to reflect the error, the client must be processed in accordance with the error messages in other HTTP specification.

The example below includes the attributes within the Permission resource model that must be included in the query response.

REQUEST

```
GET https://{serverRoot}/usersandroles/v1/permission?user.id=u123
Content-type: application/json
```

RESPONSE

```
200
Content-Type: application/json
[{
  "id":"Prms123",
  "href":"endpoint/usersandroles/v1/permission/Prms123",
  "date": "2016-03-25T12:00:00",
  "period":
    {
     "startDateTime":"2016-03-25T12:00:00",
     "endDateTime": "2017-03-25T12:00:00"
    },
  "user":
   {
     "id"="u123",
     "href": "http://server:port/PartyManagement/individual/u123"
   },
  "granter":
   {
     "id"="u987",
     "href": "http://server:port/PartyManagement/individual/u987"
   },
  "privilege":
   [
    {
     "manageableAsset":
      {
       "id"= "Asset987",
       "entityType"="IPTV license"
      },
     "action":"R&W"
    },
    {
     "manageableAsset":
      {
       "id"="Asset123",
       "entityType"="mobile line"
      },
     "action":"R/O"
    }
   ]
}]
```

The example below includes the request to retrieve Permissions impacting a given granter.

REQUEST

```
GET https://{serverRoot}/usersandroles/v1/permission?granter.id=u987
Content-type: application/json
```

RESPONSE

```
200
Content-Type: application/json
[{
  "id":"Prms123",
  "href":"endpoint/usersandroles/v1/permission/Prms123",
  "date": "2016-03-25T12:00:00",
  "period":
    {
     "startDateTime":"2016-03-25T12:00:00",
     "endDateTime": "2017-03-25T12:00:00"
    },
  "user":
   {
     "id"="u123",
     "href": "http://server:port/PartyManagement/individual/u123",
     "name": "John Doe"
   },
  "granter":
   {
     "id"="u987",
     "href": "http://server:port/PartyManagement/individual/u987",
     "name": "Jim Grants"
   },
  "privilege":
   [
    {
     "manageableAsset":
      {
       "id"= "Asset987",
       "entityType"="IPTV license"
      },
     "function":"Netflix configuration",
     "action":"R&W"
    },
    {
     "manageableAsset":
      {
       "id"= "Asset987",
       "entityType"="IPTV license"
      },
     "function":"Sport basic package",
     "action":"watch"
    },
    {
     "manageableAsset":
      {
       "id"="Asset123",
       "entityType"="mobile line"
      },
     "function":"last calls",
     "action":"R/O"
    }
   ]
}]
```

The example below includes the request to retrieve Permissions impacting a given manageable asset.

REQUEST

```
GET
https://{serverRoot}/usersandroles/v1/permission?manageabelAsset.id=a123
Content-type: application/json
```

RESPONSE

```
200
Content-Type: application/json
[{
  "id":"Prms123",
  "href":"endpoint/usersandroles/v1/permission/Prms123",
  "date": "2016-03-25T12:00:00",
  "period":
    {
     "startDateTime":"2016-03-25T12:00:00",
     "endDateTime": "2017-03-25T12:00:00"
    },
  "user":
   {
     "id"="u123",
     "href": "http://server:port/PartyManagement/individual/u123",
     "name": "John Doe"
   },
  "granter":
   {
     "id"="u987",
     "href": "http://server:port/PartyManagement/individual/u987",
     "name": "Jim Grants"
   },
  "privilege":
   [
    {
     "manageableAsset":
      {
       "id"="a123",
       "entityType"="mobile line"
      },
     "function":"last calls",
     "action":"R/O"
    }
   ]
}]
```

POST /usersandroles/v1/permission

Description:

The Application invokes this operation to create a new permission granting an individual access to manage assets of another individual.

Behavior:

Status Code	Description
201	Successful permission created
202	Accepted - The request has been accepted for processing, but the report cannot be generated immediately and will be generated at a later stage.
400	Request Error
500	The server encountered an unexpected condition which prevented it from fulfilling the request
Other	The server may use other HTTP error status codes to reflect the error, the client must be processed in accordance with the error messages in other HTTP specification.

The example below includes the attributes within the Permission entity resource model that are mandatory to be included in the request when creating a new resource in the server.

REQUEST

```
POST https://{serverRoot}/usersandroles/v1/permission
```

```
Content-type: application/json
{
  "period":
    {
     "startDateTime":"2016-03-25T12:00:00",
     "endDateTime": "2017-03-25T12:00:00"
    },
  "user":
   {
     "id"="u123"
   },
  "privilege":
   [
    {
     "manageableAsset":
      {
       "id"= "Asset987",
       "entityType"="IPTV license"
      },
     "action":"R&W"
    },
    {
     "manageableAsset":
      {
       "id"= "Asset987",
       "entityType"="IPTV license"
      },
     "function":"Sport basic package",
     "action":"watch"
    },
    {
     "manageableAsset":
      {
       "id"="Asset123",
       "entityType"="mobile line"
      },
     "action":"R/O"
    }
   ]
}
```

RESPONSE

```
201
Content-Type: application/json
Location: https://{serverRoot}/usersandroles/v1/permission/Prms123


      Response is not required to include a BODY with the contents of
      the Permission resource created, but if included it must be filled
      with at least the mandatory parameters.
```

The example below includes the case where the Permission refers to a set of preconfigured entitlements via allocating a user role to another user in relation to a given manageable asset.

REQUEST

```
POST https://{serverRoot}/usersandroles/v1/permission
Content-type: application/json
{
  "period":
    {
     "startDateTime":"2016-03-25T12:00:00",
     "endDateTime": "2017-03-25T12:00:00"
    },
  "user":
   {
     "id"="u123"
   },
  "assetUserRole":
   [
    {
     "manageableAsset":
      {
       "id"= "Asset987",
       "entityType"="IPTV license"
      },
     "userRole":
      {
       "id"= "UR001",
       "role"="owner"
      }
    }
   ]
}
```

RESPONSE

```
201
Content-Type: application/json
Location: https://{serverRoot}/usersandroles/v1/permission/Prms123


      Response is not required to include a BODY with the contents of
      the Permission resource created, but if included it must be filled
      with at least the mandatory parameters.
```

GET /usersandroles/v1/permission/{permissionId}

Description:

The Application invokes this operation to retrieve the contents of one specific permission.

Behavior:

Status Code	Description
200	Resource information was returned successfully
400	Request Error
500	The server encountered an unexpected condition which prevented it from fulfilling the request
Other	The server may use other HTTP error status codes to reflect the error, the client must be processed in accordance with the error messages in other HTTP specification.

The example below includes the attributes within the Permission resource model that must be included in the query response.

REQUEST

```
GET https://{serverRoot}/usersandroles/v1/permission/Prms123
Content-type: application/json
```

RESPONSE

```
{
  "id":"Prms123",
  "href":"endpoint/usersandroles/v1/permission/Prms123",
  "date": "2016-03-25T12:00:00",
  "description": "this is the permission to access TV and mobile line",
  "period":
    {
     "startDateTime":"2016-03-25T12:00:00",
     "endDateTime": "2017-03-25T12:00:00"
    },
  "user":
   {
     "id"="u123",
     "href": "http://server:port/PartyManagement/individual/u123",
     "name": "John Doe"
   },
  "granter":
   {
     "id"="u987",
     "href": "http://server:port/PartyManagement/individual/u987",
     "name": "Jim Grants"
   },
  "privilege":
   [
    {
     "manageableAsset":
      {
       "id"= "Asset987",
       "entityType"="IPTV license"
      },
     "function":"Netflix configuration",
     "action":"R&W"
    },
    {
     "manageableAsset":
      {
       "id"= "Asset987",
       "entityType"="IPTV license"
      },
     "function":"Sport basic package",
     "action":"watch"
    },
    {
     "manageableAsset":
      {
       "id"="Asset123",
       "entityType"="mobile line"
      },
     "function":"last calls",
     "action":"R/O"
    }
   ]
}
```

PATCH /usersandroles/v1/permission/{permissionId}

This operation is optional to be supported in this API

Description:

The Application invokes this operation to partially update the information about a permission previously created.

The elements that are expected to be modified in the Permission resource are the validity and the action allowed over a given manageable asset or function.

Behavior:

To Be Defined.

PUT /usersandroles/v1/permission/{permissionId}

This operation is optional to be supported in this API

Description:

The Application invokes this operation to completely update the information about a permission previously created.

The elements that are expected to be modified in the Permission resource are the validity and the action allowed over a given manageable asset or function.

Notice that the PUT method is intended to modify completely the resource impacted, meaning that optional values that are not included in the request may be erased in the server after updating, and will not keep the previous value stored. Behaviour of the server on optional values not included is undefined.

Behavior:

To Be Defined.

GET /usersandroles/v1/role

Description:

The Application invokes this operation to retrieve the user roles stored in the server.

The request could include a filter to retrieve only the roles that match specific criteria (e.g.: created within a given date range).

Behavior:

Status Code	Description
200	User roles information was returned successfully
400	Request Error
500	The server encountered an unexpected condition which prevented it from fulfilling the request
Other	The server may use other HTTP error status codes to reflect the error, the client must be processed in accordance with the error messages in other HTTP specification.

The example below includes the attributes within the UserRole resource model that must be included in the query response.

REQUEST

```
GET https://{serverRoot}/usersandroles/v1/role
Content-type: application/json
```

RESPONSE

```
200
Content-Type: application/json
[
{
  "id":"UR001",
  "href":"endpoint/usersandroles/v1/role/UR001",
  "involvementRole":"owner",
  "entitlement":
   [
    {
     "function":"all",
     "action":"R&W"
    }
   ]
},
{
  "id":"UR001",
  "href":"endpoint/usersandroles/v1/role/UR001",
  "involvementRole":"member",
  "entitlement":
   [
    {
     "function":"all",
     "action":"R/O"
    }
   ]
},
{
  "id":"UR123",
  "href":"endpoint/usersandroles/v1/role/UR123",
  "involvementRole":"configure IPTV and watch news",
  "entitlement":
   [
    {
     "function":"Netflix configuration",
     "action":"R&W"
    },
    {
     "function":"Sport basic package",
     "action":"watch"
    }
   ]
}
]
```

POST /usersandroles/v1/role

Description:

The Application invokes this operation to create a new user role.

Behavior:

Status Code	Description
201	User role created
202	Accepted - The request has been accepted for processing, but the report cannot be generated immediately and will be generated at a later stage.
400	Request Error
500	The server encountered an unexpected condition which prevented it from fulfilling the request
Other	The server may use other HTTP error status codes to reflect the error, the client must be processed in accordance with the error messages in other HTTP specification.

The example below includes the attributes within the UserRole entity resource model that are mandatory to be included in the request when creating a new resource in the server.

REQUEST

```
POST https://{serverRoot}/usersandroles/v1/role
Content-type: application/json
{
  "involvementRole":"configure IPTV and watch news",
  "entitlement":
   [
    {
     "function":"Netflix configuration",
     "action":"R&W"
    },
    {
     "function":"Sport basic package",
     "action":"watch"
    }
   ]
}
```

RESPONSE

```
201
Content-Type: application/json
Location: https://{serverRoot}/usersandroles/v1/role/UR123


      Response is not required to include a BODY with the contents of
      the Permission resource created, but if included it must be filled
      with at least the mandatory parameters.
```

GET /usersandroles/v1/role/{roleId}

Description:

The Application invokes this operation to retrieve the contents of one specific user role.

Behavior:

Status Code	Description
200	Resource information was returned successfully
400	Request Error
500	The server encountered an unexpected condition which prevented it from fulfilling the request
Other	The server may use other HTTP error status codes to reflect the error, the client must be processed in accordance with the error messages in other HTTP specification.

The example below includes the attributes within the UserRole resource model that must be included in the query response.

REQUEST

```
GET https://{serverRoot}/usersandroles/v1/role/UR123
Content-type: application/json
```

RESPONSE

```
{
  "id":"UR123",
  "href":"endpoint/usersandroles/v1/role/UR123",
  "involvementRole":"configure IPTV and watch news",
  "entitlement":
   [
    {
     "function":"Netflix configuration",
     "action":"R&W"
    },
    {
     "function":"Sport basic package",
     "action":"watch"
    }
   ]
}
```

PATCH /usersandroles/v1/role/{roleId}

This operation is optional to be supported in this API

Description:

The Application invokes this operation to partially update the information about a user role previously created.

The element that is expected to be modified in the User Role resource is the action allowed over a given function.

Behavior:

To Be Defined.

PUT /usersandroles/v1/role/{roleId}

This operation is optional to be supported in this API

Description:

The Application invokes this operation to completely update the information about a user role previously created.

The element that is expected to be modified in the User Role resource is the action allowed over a given function.

Notice that the PUT method is intended to modify completely the resource impacted, meaning that optional values that are not included in the request may be erased in the server after updating, and will not keep the previous value stored. Behaviour of the server on optional values not included is undefined.

Behavior:

To Be Defined.

API NOTIFICATION

It is assumed that the Pub/Sub uses the Register and UnRegister mechanisms described in the REST Guidelines reproduced below.

REGISTER LISTENER

POST /hub

Description

Sets the communication endpoint address the service instance must use to deliver information about its health state, execution state, failures and metrics. Subsequent POST calls will be rejected by the service if it does not support multiple listeners. In this case DELETE /api/hub/{id} must be called before an endpoint can be created again.

Behavior

Returns HTTP/1.1 status code 204 if the request was successful.

Returns HTTP/1.1 error status code if request is not successful.

Usage Samples

Here's an example of a request for registering a listener.

Request

```
POST /api/hub
Accept: application/json

{"callback": "http://in.listener.com"}
```

Response

```
201
Content-Type: application/json
Location: /api/hub/42

{"id":"42","callback":"http://in.listener.com","query":null}
```

UNREGISTER LISTENER

DELETE /hub/{id}

Description

Clears the communication endpoint address that was set by creating the Hub.

Behavior

Returns HTTP/1.1 status code 204 if the request was successful.

Returns HTTP/1.1 status code 404 if the resource is not found.

Usage Samples

Here's an example of a request for un-registering a listener.

Request

```
DELETE /api/hub/42
Accept: application/json
```

Response

```
204
```

PUBLISH EVENT TO LISTENER

POST /client/listener

Description

Clears the communication endpoint address that was set by creating the Hub.

Provides to a registered listener the description of the event that was raised. The /client/listener url is the callback url passed when registering the listener.

Behavior

Returns HTTP/1.1 status code 201 if the service is able to set the configuration.

Usage Samples

Here's an example of a notification received by the listener. In this example "EVENT TYPE" should be replaced by one of the notification types supported by this API (see Notification resources Models section) and EVENT BODY refers to the data structure of the given notification type.

Request

```
POST /client/listener
Accept: application/json

{
   "eventId": "111",
   "eventType": " EVENT_TYPE ",
   "event": {
            EVENT BODY as described in event model section
    }
}
```

Response
201

For detailed examples on the general TM Forum notification mechanism, see the TMF REST Design Guidelines.

ACKNOWLEDGMENTS

RELEASE HISTORY

Release Number	Date	Release led by:	Description
Release 0.1	21/March/2017		First Release of Draft Version of the Document.
Release 0.3	08/June/2017		Draft generated after official review.
Release 17.0.1 Version 0.3.1	21/November/2017	Adrienne Walcott	Updated to reflect TM Forum Approved Status