



User Update REST API Technical Documentation

Version 1.1

Contents

1	Document Change Management.....	3
2	Introduction.....	4
3	Authentication.....	5
3.1	Login Service.....	5
3.2	Logout Service.....	6
4	Service Proxies.....	7
4.1	Operation Service Proxy.....	8
4.1.1	Stateful Operation Service Proxy.....	8
4.1.2	WS Operation execute.....	8
4.1.3	Stateless Operation Service Proxy.....	8
4.1.4	WS Operation execute.....	8
5	User Services.....	9
5.1	Create user.....	9
5.2	User Search.....	9
5.3	User Search At.....	9
5.4	Update user data.....	10
5.4.1	Fields description.....	13
5.5	Update user data (PME).....	15
5.5.1	Fields description.....	16
5.6	Update user data (Attributes).....	18
5.6.1	Fields description.....	19
6	Auxiliary Services to get data.....	21
6.1	Get PME.....	21

1 Document Change Management

Release	Date	Author	Comments
1.0	06/2020	Ricardo Fernandes	First release
1.1	12/2020	Ricardo Fernandes	Add the services of Create/Search/Search At Add start page option on Update user

2 Introduction

Several REST web services are provided allowing tenant and customers to interact with Triskell virtually from any environment/platform. This document describes how to manage the update of the user parameters and data.



Usually, Web services are designed to manage small set of data.

If you are not sure they will cover your requirements please contact us at :
support@triskellsoftware.com.

In this document there are some examples about how to send information to Triskell using Postman Google plugin. You can get postman free [here](#). Then you just need to import the attached file "UserUpdate.postman_collection.json" and the file "*NextRelease.postman_environment.json*". The last file is used to set environment variables, so it can be easily tested on different environments.

It's very important to test your code on the Nextrelease environment before delivering it on the production environment (<https://nextrelease.triskellsoftware.com/triskell/>). In this document examples given are using Localhost as testing environment.



Some internal id's can be different between testing and production environments. Please review the hardcoded id's before delivering it on production environment.

3 Authentication

From the authentication point of view, there are two kinds of web services on Triskell: stateful and stateless.

Most of the REST services provided are *Stateful*, which means that interacting with them requires a login first. On the other hand, *Stateless* services will do a user authentication on every call.

Stateless services are intended to be used from clients not capable of managing a session cookie to maintain a dialog with the server.



In this document, only Stateless services will be denoted, most of the services being Stateful by default.

Two services are provided to log in and log out of Triskell.

Stateless services must provide their authentication parameters as HTTP Headers on every service call.

3.1 Login Service

The login service must be invoked before any other stateful Triskell web service call. It receives a user identifier and a password as URL parameters.

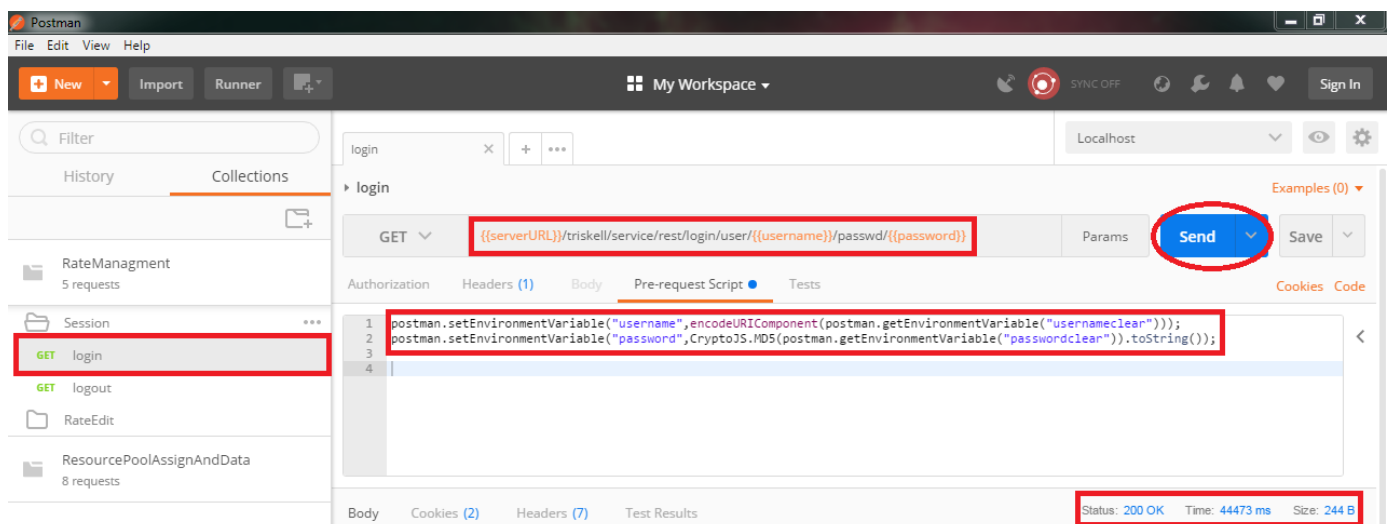
URL	https:// SERVER /triskell/service/rest/login/user/{userId}/passwd/{pass}
HTTP Method	GET
URL Parameters	{ userId } : String containing user identifier as 'user@tenant.com' URI encoded { pass } : String containing a Base64 encoded MD5 hash of the password

Note: the user used to make the login need have the right privileges/roles in the Triskell to use/change the data. Otherwise it will not be possible to use/change the data and will return an error of missing privileges.



When login is successful it returns a HTTP response code '200 – OK'
An authentication failure will return HTTP response code '401 – UNAUTHORIZED'

Example:

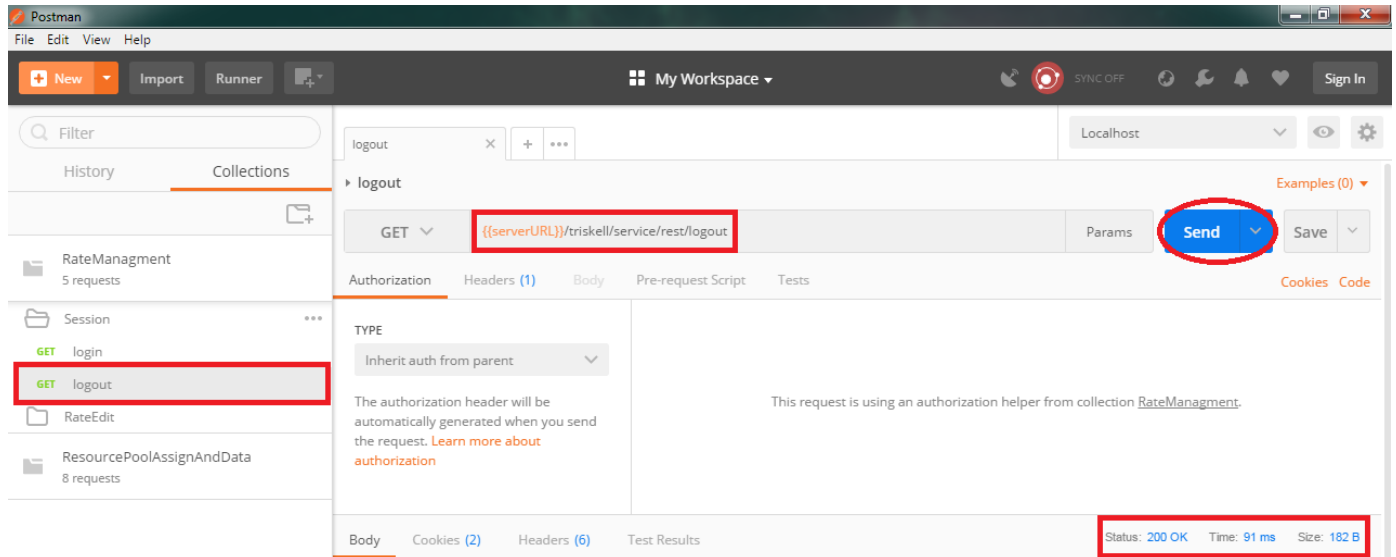


3.2 Logout Service

Logout service provided for closing the session at the server.

URL	URL: <code>https:// SERVER /triskell/service/rest/logout</code>
HTTP Method	GET

Example:



Don't forget to close your session when finishing your REST conversation

4 Service Proxies

Service Proxies are a simple way of executing remote procedure calls Operation services.

Requests and responses are exchanged as JSON objects with the clients, to enhance easiness and interoperability.

The proxy input object DataRequest, is compound of the following properties:

- Id : numerical identifier
- Params: A simple JSON object containing only primitive type properties.
- Objects: Array of JSON objects of any type.

These properties must be present into the object being it used or not.

Example of DataRequest:

```
{
  'id' : 1 ,
  'params' : {
    "param_1": "A",
    "param_2": 2
  },
  'objects' : [
    {"NAME": "OBJECT_1"},
    {"NAME": "OBJECT_2"}
  ]
}
```

The proxy output object Result, is compound of the following properties:

- success: Boolean indicating successfulness of the request.
- message: Error description.
- resultType: Numerical identifier, error severity (1-OK, 2–Warning, 3-Error)
- data: a String or a JSON object
- id: a numerical identifier
- authash: session identifier
- executime: service execution time in nanoseconds
- i18NParams and i18NMessageId: used on internationalized error message

Example of DataResult:

```
{
  "authash": "dd06d7c94cd9a73f62b45c5b54247bce",
  "executime": 10997027,
  "success": true,
  "message": null,
  "resultType": 1,
  "data": "Hello world !!!",
  "id": 0,
  "i18NParams": [],
  "i18NMessageId": ""
}
```

4.1 Operation Service Proxy

This is a Web Service to execute operations on a Triskell OperationService.

An OperationService groups related functionalities allowing to access then by his Operation Name.

OperationServiceProxy has two implementations, stateful and stateless.

4.1.1 Stateful Operation Service Proxy

Any stateful service requires a Login to be done before submitting any request to it.

4.1.2 WS Operation execute

Execute a service operation call. DataRequest is sent attached to the body of the HTTP request.

URL	URL: https:// SERVER /triskell/service/rest/proxy/operation/execute/{serviceName}/{operationName}
HTTP Method	POST
Content-Type Header	application/json
URL Parameters	{serviceName} : String containing the name of the service to be called {operationName} : String containing the Operation name

4.1.3 Stateless Operation Service Proxy

This web service does not require doing a previous Login on Triskell, a user authentication is done on every request.

4.1.4 WS Operation execute

Execute a service operation call. DataRequest is sent as a URL parameter.

URL	https:// SERVER /triskell/service/rest/proxy/operation/execute/{serviceName}/{operationName}/{payload}
HTTP Method	GET
Content-Type Header	X-Account-Name Header: Optional, username@tenantdomain X-API-Key Header: Optional
URL Parameters	{serviceName} : String containing the name of the service to be called {operationName} : String containing the Operation name {payload} : JSON DataRequest Object encoded in Base64

5 User Services

In this section, we describe the main operations available to handle the users via Rest Services in Triskell using the existing Stateful services for the most basic use of case. In this document, we are using examples from a standard configuration at test.com.

5.1 Create user

This web service allows create a new user in Triskell.

This service is already explained on another document that handle the service calls to do the creation of the user.

Please refer to the document “Triskell-REST API.docx”, that can be found [here](#).

5.2 User Search

This web service search for a user in Triskell.

This service is already explained on another document that handle the service calls to do the search of the user.

Please refer to the document “Triskell-REST API.docx”, that can be found [here](#).

5.3 User Search At

This web service search for a user in Triskell with were clause.

This service is already explained on another document that handle the service calls to do the search of the user.

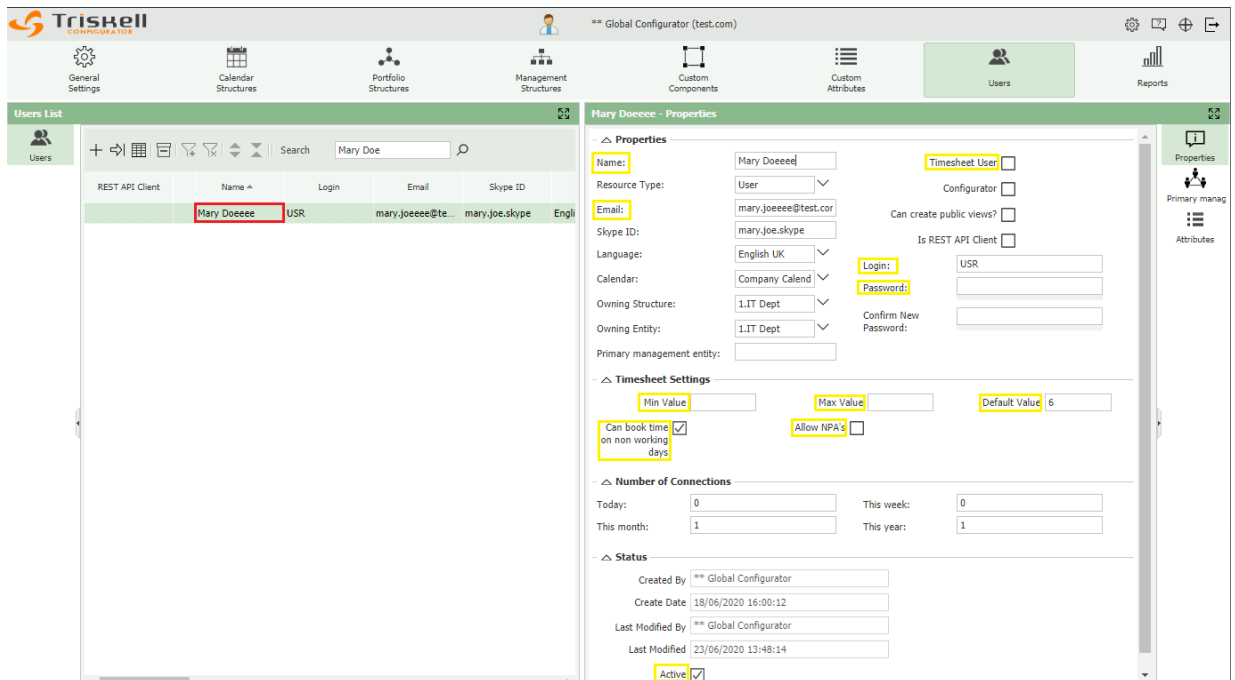
Please refer to the document “Triskell-REST API.docx”, that can be found [here](#).

5.4 Update user data

To use this call, the user logged in do not need to have the Configurator role in Triskell, also we need get some information from your instance to make it work properly.

What information do we need to edit a User parameters/data?

- User to edit the parameters: "Mary Doe"e"
- Parameters available to be updated:
 - Name
 - Email
 - Login
 - Password
 - Active
 - Timesheet user
 - Can book time on non-working days
 - Allow NPA's
 - Min Value
 - Max Value
 - Default Value
 - Primary management entity
 - Attributes []



Triskell CONFIGURATOR ** Global Configurator (test.com)

General Settings | Calendar Structures | Portfolio Structures | Management Structures | Custom Components | Custom Attributes | **Users** | Reports

Users List

REST API Client	Name	Login	Email	Skype ID	Language
	Mary Doe	USR	mary.joe@te...	mary.joe.skype	Engl

Mary Doe - Properties

Properties

Name: Mary Doe Timesheet User ☐

Resource Type: User

Email: mary.joe@te... Configurator ☐

Skype ID: mary.joe.skype Can create public views? ☐

Language: English UK Login: USR

Calendar: Company Calend Password:

Owning Structure: 1.IT Dept Confirm New Password:

Owning Entity: 1.IT Dept

Primary management entity:

Timesheet Settings

Min Value Max Value Default Value 6

Can book time on non working days ☒ Allow NPA's ☐

Number of Connections

Today: 0 This week: 0

This month: 1 This year: 1

Status

Created By: ** Global Configurator

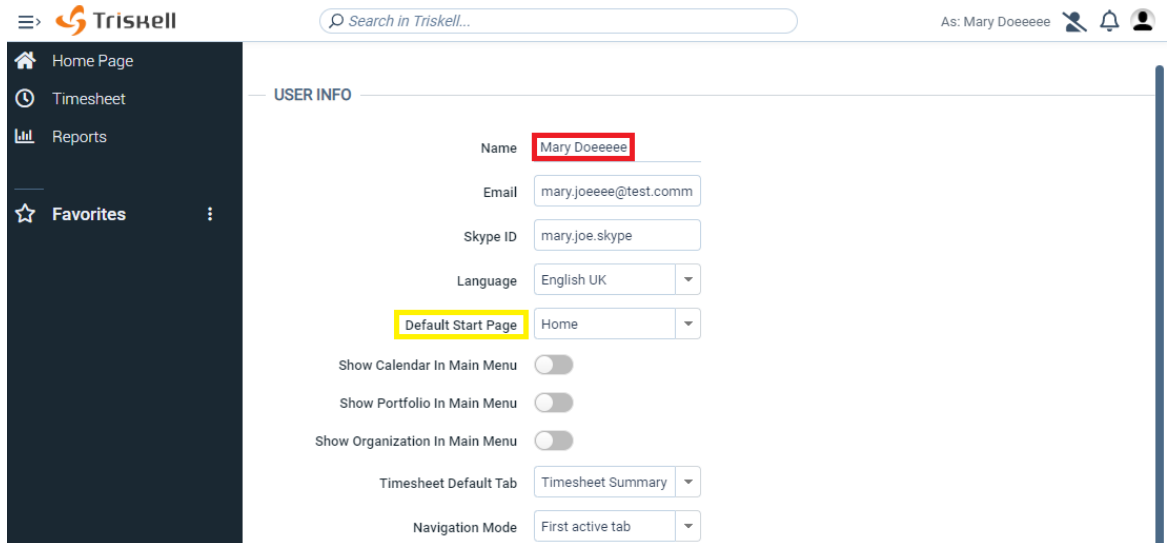
Create Date: 18/06/2020 16:00:12

Last Modified By: ** Global Configurator

Last Modified: 23/06/2020 13:48:14

Active ☒

- Default Start Page

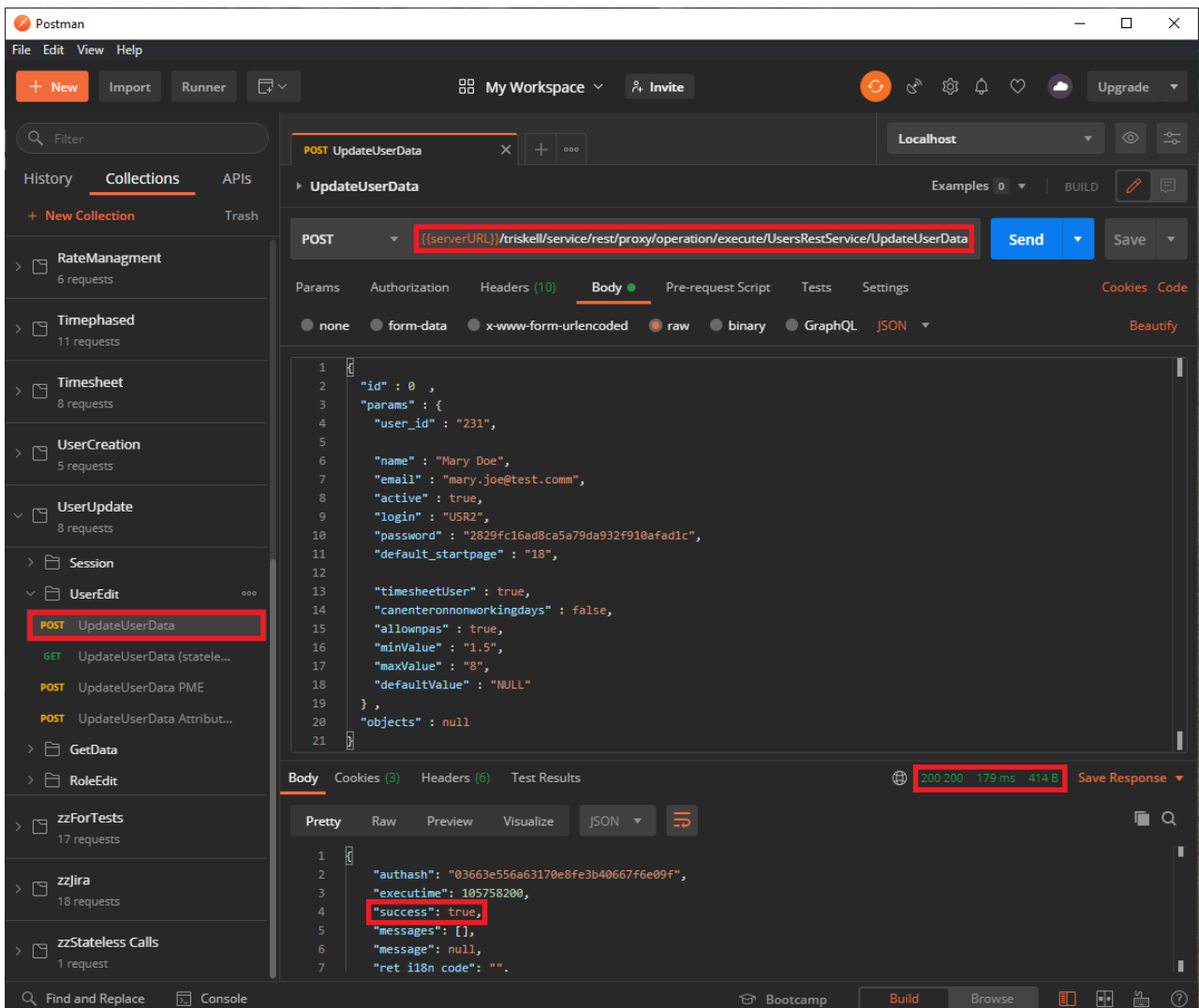


The screenshot shows the Triskell user profile page. The left sidebar contains navigation links: Home Page, Timesheet, Reports, and Favorites. The main content area is titled 'USER INFO' and contains the following fields:

- Name: Mary Doe (highlighted with a red box)
- Email: mary.joe@test.comm
- Skype ID: mary.joe.skype
- Language: English UK (dropdown)
- Default Start Page: Home (dropdown, highlighted with a yellow box)
- Show Calendar In Main Menu: ☐
- Show Portfolio In Main Menu: ☐
- Show Organization In Main Menu: ☐
- Timesheet Default Tab: Timesheet Summary (dropdown)
- Navigation Mode: First active tab (dropdown)

We need to extract all this information from Triskell. Some information can be taken directly from your Triskell instance using the UI and, in some cases, we need to make a request using an operational service to get data from a stored selector report.

This is a request example:



The screenshot shows a Postman REST client interface. The left sidebar displays a list of collections, with 'UserUpdate' selected. The main area shows a POST request to the endpoint `{{serverURL}}/triskell/service/rest/proxy/operation/execute/UsersRestService/UpdateUserData` (highlighted with a red box). The request body is in JSON format:

```

1 {
2   "id" : 0 ,
3   "params" : {
4     "user_id" : "231",
5
6     "name" : "Mary Doe",
7     "email" : "mary.joe@test.comm",
8     "active" : true,
9     "login" : "USR2",
10    "password" : "2829fc16ad8ca5a79da932f910afad1c",
11    "default_startpage" : "18",
12
13    "timesheetUser" : true,
14    "canenteronnonworkingdays" : false,
15    "allownpas" : true,
16    "minValue" : "1.5",
17    "maxValue" : "8",
18    "defaultValue" : "NULL"
19  } ,
20   "objects" : null
21 }

```

The response is shown in the bottom panel, with a status of 200 OK (200 200, 179 ms, 414 B, highlighted with a red box). The response body is in JSON format:

```

1 {
2   "authash" : "03663e556a63170e8fe3b40667f6e09f",
3   "executetime" : 105758200,
4   "success" : true,
5   "messages" : [],
6   "message" : null,
7   "ret i18n code" : ""
8 }

```

This is the example content request:

```
{
  "id" : 0 ,
  "params" : {
    "user_id" : "231",

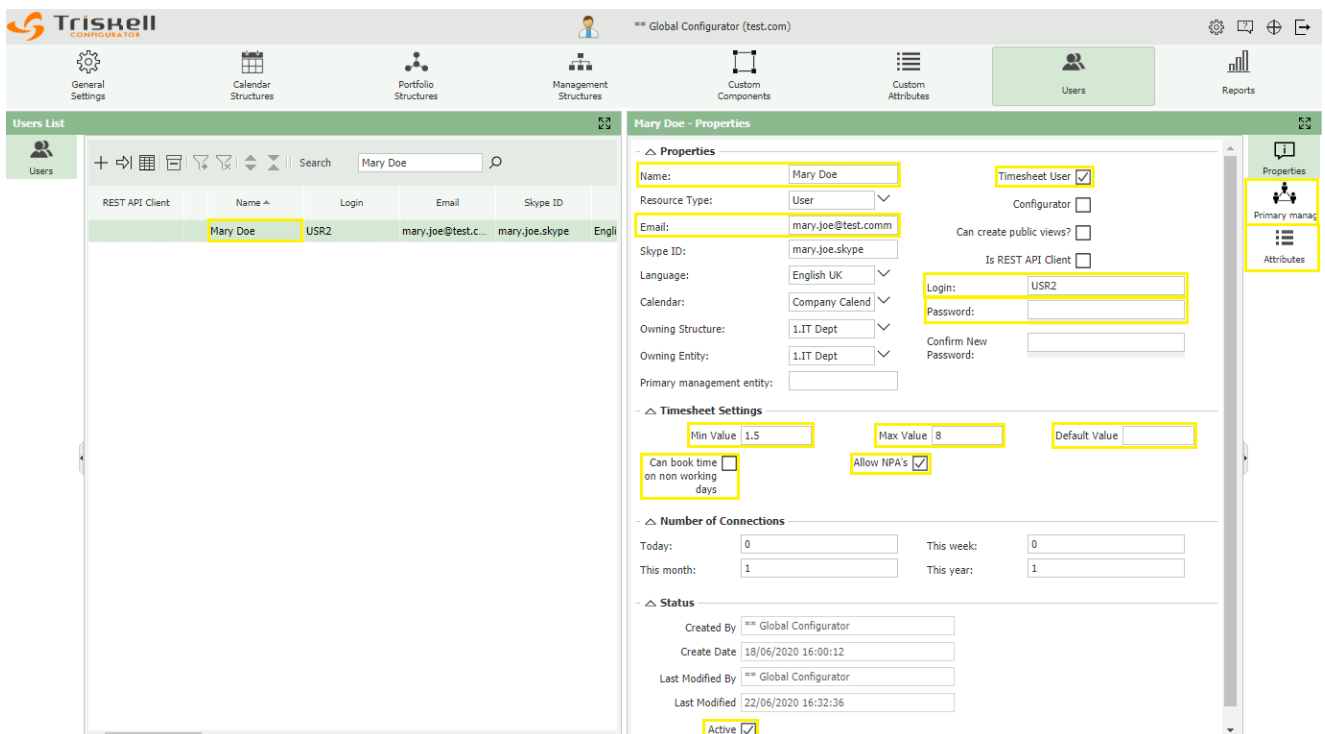
    "name" : "Mary Doe",
    "email" : "mary.joe@test.comm",
    "active" : true,
    "login" : "USR2",
    "password" : "2829fc16ad8ca5a79da932f910afad1c",
    "default_startpage" : "18",

    "timesheetUser" : true,
    "canenteronnonworkingdays" : false,
    "allownpas" : true,
    "minValue" : "1.5",
    "maxValue" : "8",
    "defaultValue" : "NULL"
  },
  "objects" : null
}
```

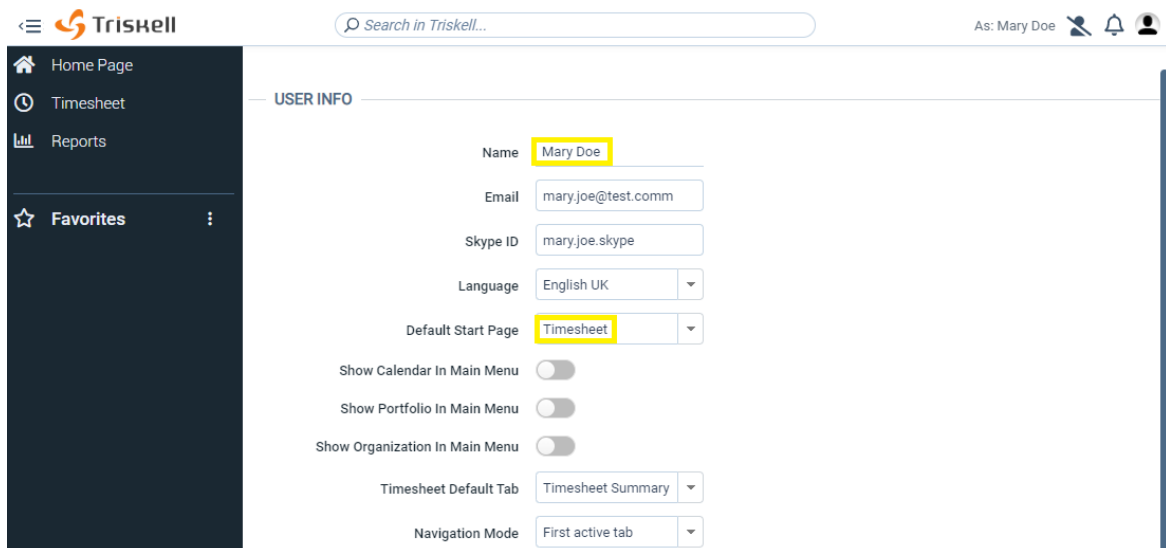


Customer can do more than one request on the same REST conversation depending on your functional needs. This feature should not be used to load big data volumes because. During the REST conversation Triskell users can get blocked processing the request, so we suggest using it with common sense in terms of data volume and scheduling. Triskell will limit the number of requests by day in the future to avoid collapsing the server, so please take this in account when developing your interfaces.

To see if the User parameters/data was edited you can check on Triskell.



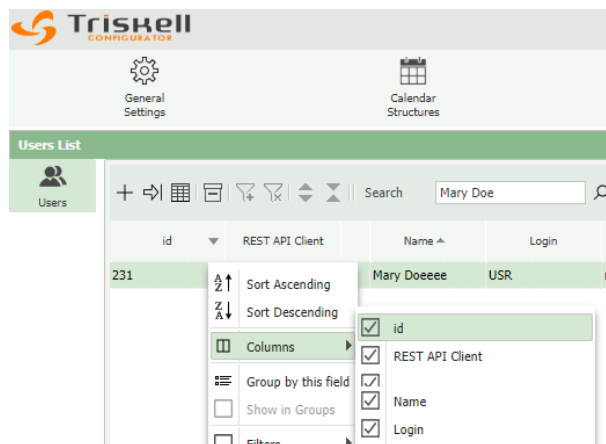
The screenshot displays the Triskell Global Configurator interface for editing user properties. The 'Mary Doe - Properties' form is active, showing various configuration options. Key fields include Name, Email, Login, Password, and checkboxes for 'Timesheet User', 'Can create public views?', and 'Is REST API Client'. The 'Timesheet Settings' section includes 'Min Value', 'Max Value', 'Default Value', and 'Allow NPA's'. The 'Number of Connections' section shows counts for Today, This week, This month, and This year. The 'Status' section shows 'Created By', 'Create Date', 'Last Modified By', and 'Last Modified'.



5.4.1 Fields description

5.4.1.1 Mandatory fields

- **user_id**: unique user identifier (Integer). In the previous example is the User "Mary Doe". Can be extracted from the UI on the configuration environment:

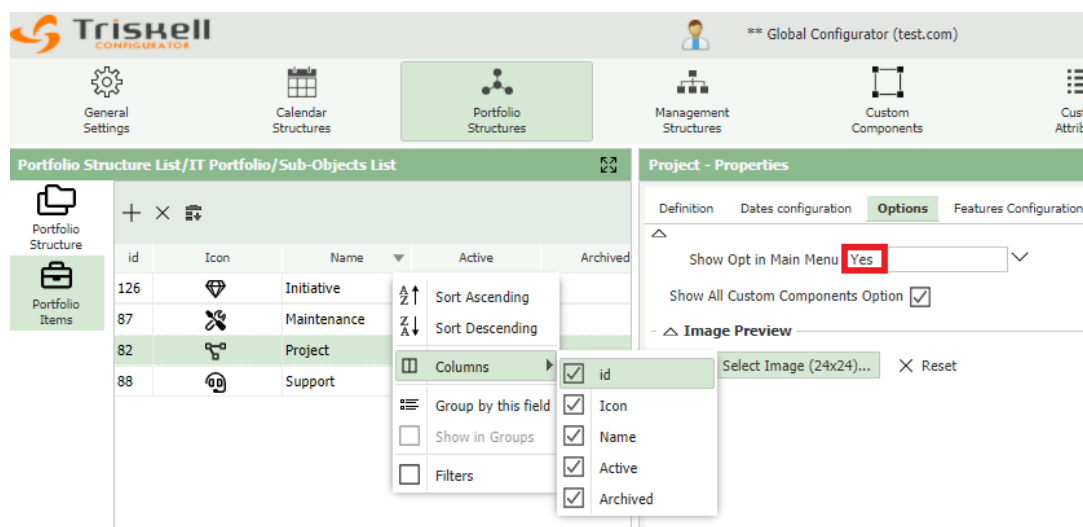


id	REST API Client	Name	Login
231	Mary Doe	USR	m

5.4.1.2 Optional fields

- **name**: the new name for the User.
- **email**: the new email for the User.
- **active**: Boolean value (true/false).
- **login**: the new login for the User.
- **password**: the new password in HEX encoded MD5 hash format, for the User.
- **timesheetUser**: Boolean value (true/false).
- **canenteronnonworkingdays**: Boolean value (true/false).
- **allownpas**: Boolean value (true/false).
- **minValue**: Big Decimal value with decimal separator "." and no thousands separator.
 - To clear this value, send "NULL".
- **maxValue**: Big Decimal value with decimal separator "." and no thousands separator.

- To clear this value, send “NULL”.
- defaultValue: Big Decimal value with decimal separator “.” and no thousands separator.
 - To clear this value, send “NULL”.
- default_startpage: unique default start page identifier (Integer). Have fixed and dynamic values.
 - Fixed:
 - 12 - Application Home Page
 - 13 - Management Structures Page
 - 14 - Portfolios Page
 - 15 - Portfolio Items Page
 - 16 - Custom Components Page
 - 17 - Management Entities Page
 - 18 - Timesheet Page
 - 19 - Calendar Page
 - 20 - Reports Page
 - Dynamic:
 - The unique Object identifier (Integer) in negative. The Objects available to be in the default start page are the ones with the option “Show Opt in Main Menu” with value YES on its definition (Ex: Project = -81, Support = -88)



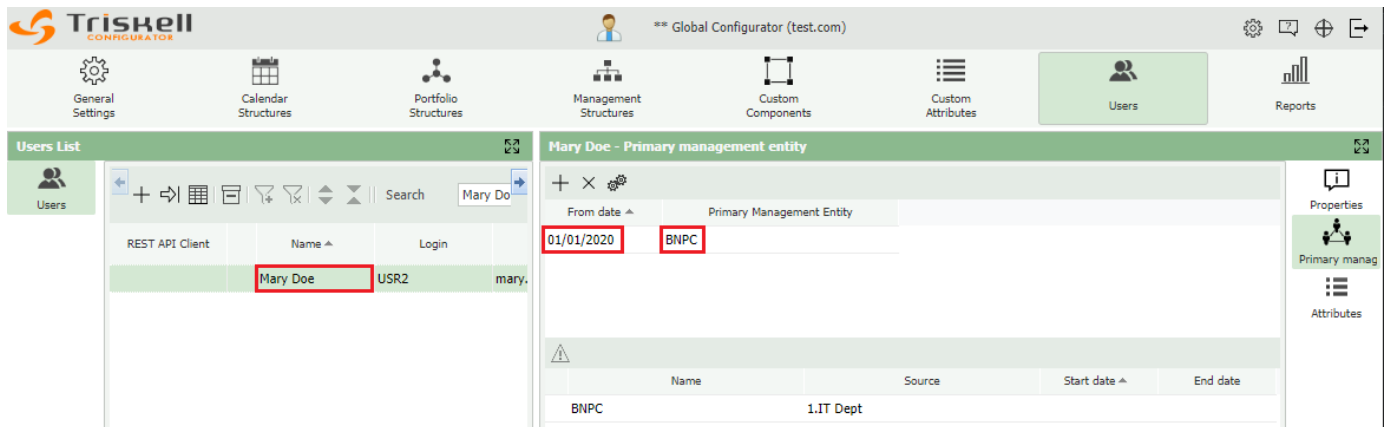
NOTE: If the value of the default_startpage is dynamic, and the user does not have any role to access to that Object, it will be loaded the Application Home Page instead.

5.5 Update user data (PME)

To use this call, the user logged in do not need to have the Configurator role in Triskell, also we need get some information from your instance to make it work properly.

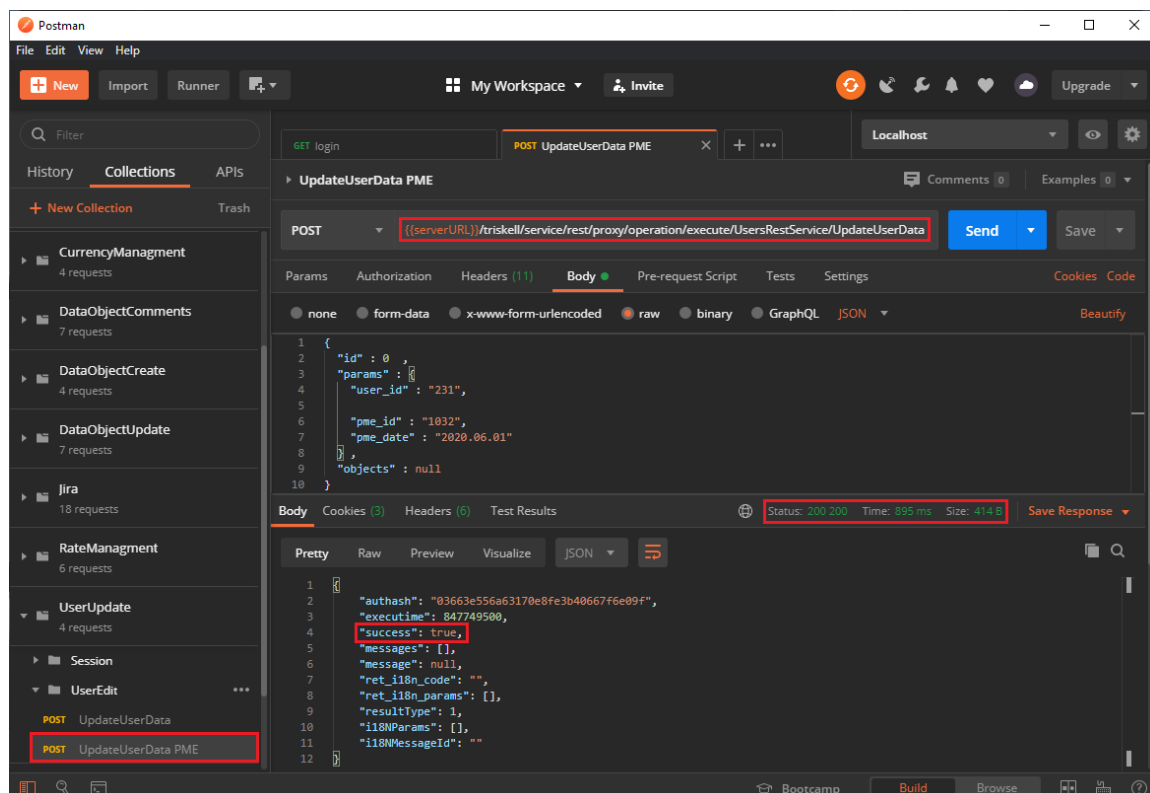
What information do we need to edit a User PME?

- User to edit the parameters: "Mary Doe"
- From date: "01/01/2020"
- Primary Management Entity: BNPC



We need to extract all this information from Triskell. Some information can be taken directly from your Triskell instance using the UI and, in some cases, we need to make a request using an operational service to get data from a stored selector report.

This is a request example:



This is the example content request:

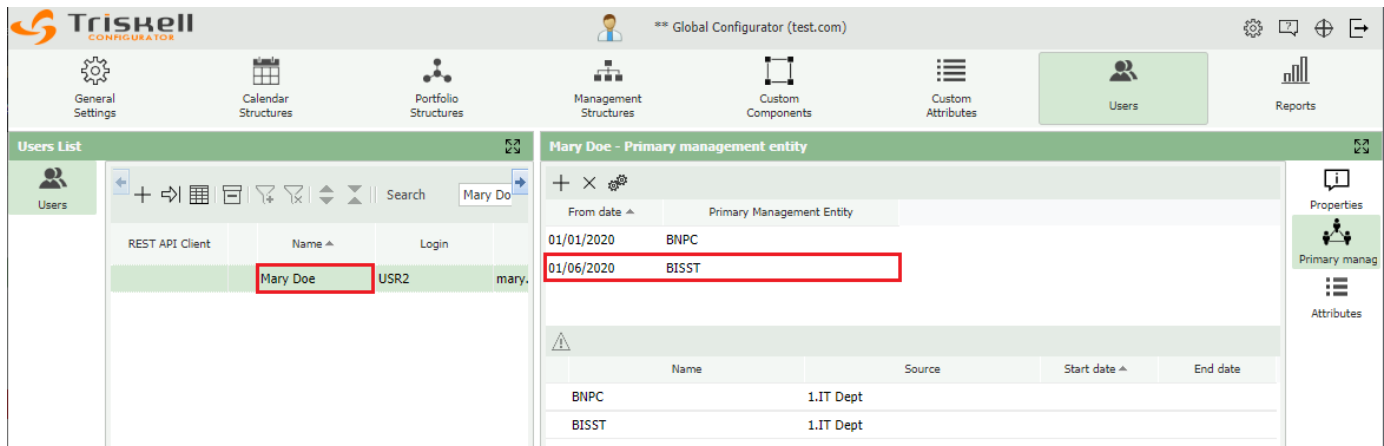
```
{
  "id" : 0 ,
  "params" : {
    "user_id" : "231",

    "pme_id" : "1032",
    "pme_date" : "2020.06.01"
  },
  "objects" : null
}
```



Customer can do more than one request on the same REST conversation depending on your functional needs. This feature should not be used to load big data volumes because. During the REST conversation Triskell users can get blocked processing the request, so we suggest using it with common sense in terms of data volume and scheduling. Triskell will limit the number of requests by day in the future to avoid collapsing the server, so please take this in account when developing your interfaces.

To see if the User PME was edited you can check on Triskell.

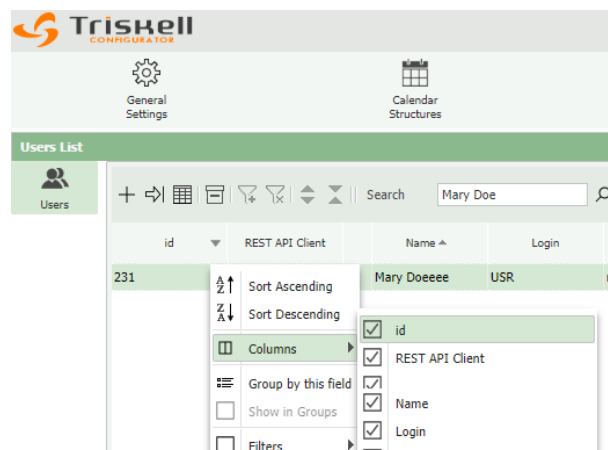


The screenshot shows the Triskell Global Configurator interface. The top navigation bar includes icons for General Settings, Calendar Structures, Portfolio Structures, Management Structures, Custom Components, Custom Attributes, Users, and Reports. The main content area is divided into two panels. The left panel, titled 'Users List', shows a table with columns for REST API Client, Name, and Login. The 'Name' column contains 'Mary Doe' and the 'Login' column contains 'USR2'. The right panel, titled 'Mary Doe - Primary management entity', shows a table with columns for From date and Primary Management Entity. The 'From date' column contains '01/01/2020' and '01/06/2020', and the 'Primary Management Entity' column contains 'BNPC' and 'BISST'. The '01/06/2020' date and 'BISST' entity are highlighted with a red box. Below this table, there is a warning icon and a table with columns for Name, Source, Start date, and End date. The 'Name' column contains 'BNPC' and 'BISST', and the 'Source' column contains '1.IT Dept' for both.

5.5.1 Fields description

5.5.1.1 Mandatory fields

- **user_id**: unique user identifier (Integer). In the previous example is the User "Mary Doe". Can be extracted from the UI on the configuration environment:



5.5.1.2 Optional fields

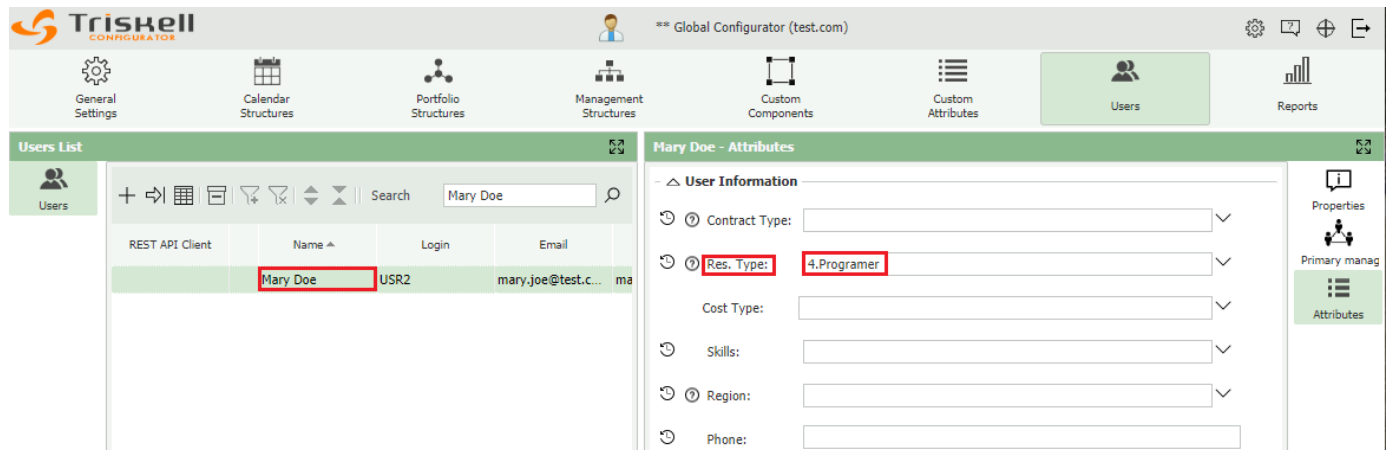
- pme_id: unique data_object identifier (Integer). In previous example is "BISST". It requires making a request to Triskell to get it.
- pme_date: unique Date in the day format (String). In previous example is "2020.06.01". The Date must be in the ISO format to work (YYYY.MM.DD), without hours, minutes, and seconds.

5.6 Update user data (Attributes)

To use this call, the user logged in do not need to have the Configurator role in Triskell, also we need get some information from your instance to make it work properly.

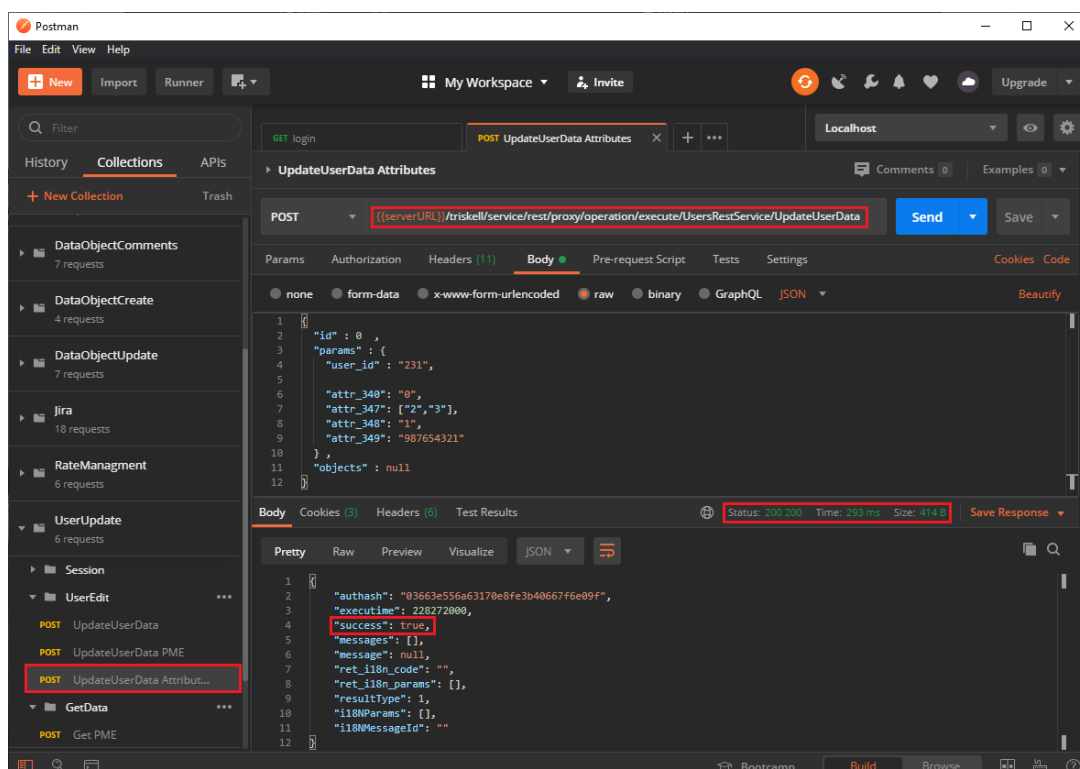
What information do we need to edit a User Attributes?

- User to edit the parameters: "Mary Doe"
- Attribute to update: "Res. Type"
- Value for the Attribute: "4.Programmer"



We need to extract all this information from Triskell. Some information can be taken directly from your Triskell instance using the UI and, in some cases, we need to make a request using an operational service to get data from a stored selector report.

This is a request example:



This is the example content request:

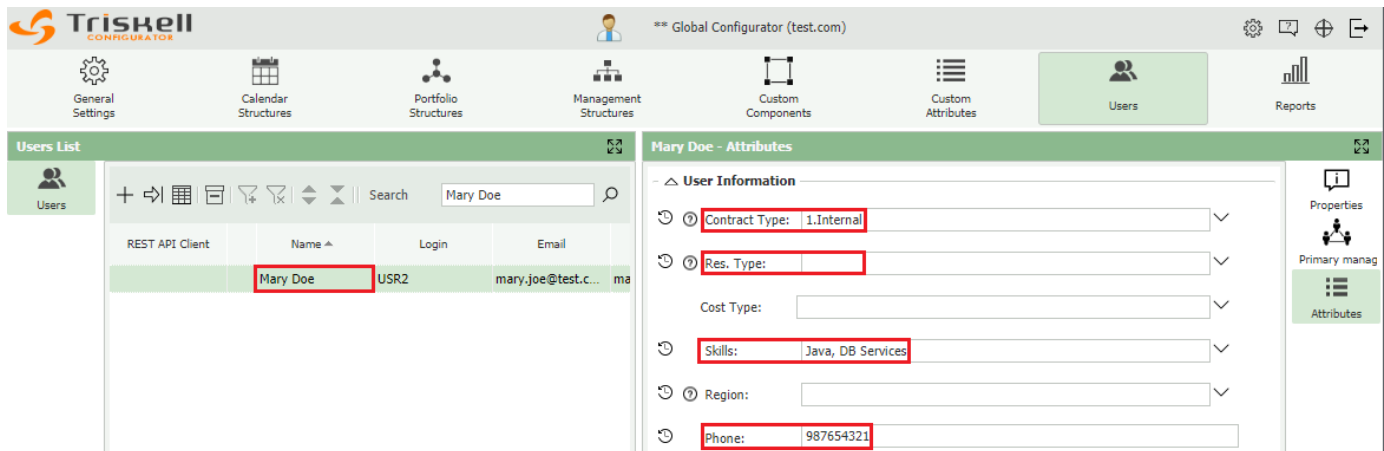
```
{
  "id" : 0 ,
  "params" : {
    "user_id" : "231",

    "attr_340": "0",
    "attr_347": ["2","3"],
    "attr_348": "1",
    "attr_349": "987654321"
  },
  "objects" : null
}
```



Customer can do more than one request on the same REST conversation depending on your functional needs. This feature should not be used to load big data volumes because. During the REST conversation Triskell users can get blocked processing the request, so we suggest using it with common sense in terms of data volume and scheduling. Triskell will limit the number of requests by day in the future to avoid collapsing the server, so please take this in account when developing your interfaces.

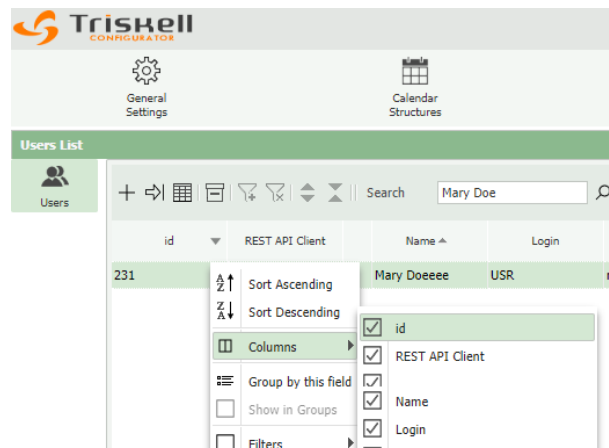
To see if the User Attributes was edited you can check on Triskell.



5.6.1 Fields description

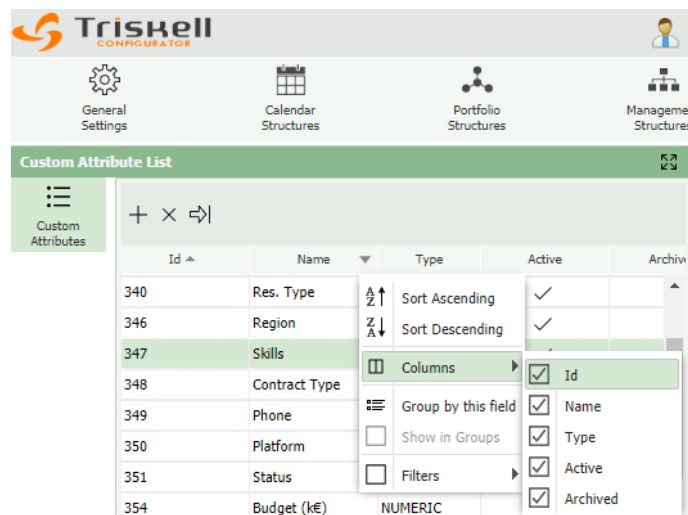
5.6.1.1 Mandatory fields

- user_id: unique user identifier (Integer). In the previous example is the User "Mary Doe". Can be extracted from the UI on the configuration environment:

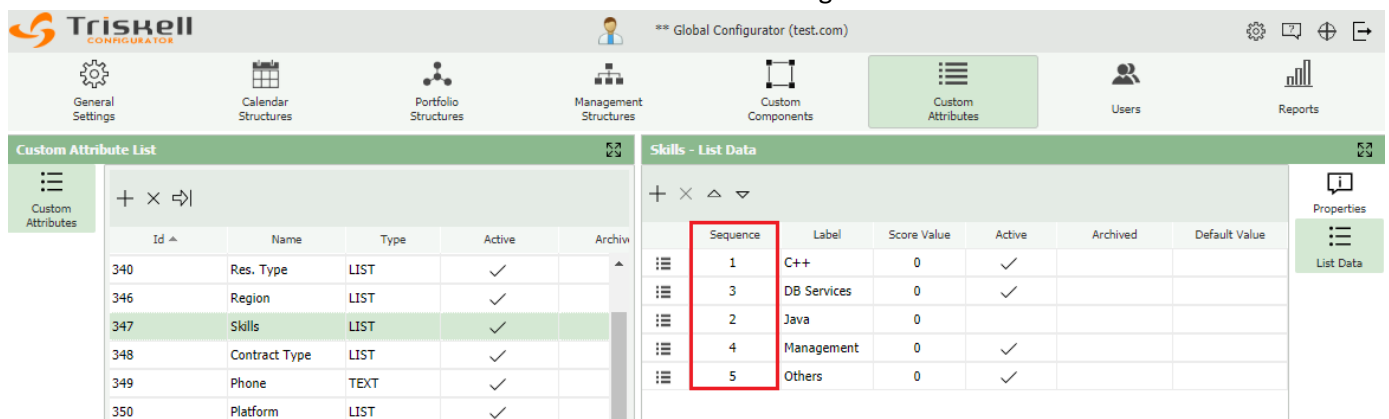


5.6.1.2 Optional fields

- `attr_XXX`: unique sequence value identifier (Integer) for the list attributes, or other value for other attribute type. In the previous example is the values of the attributes “Contract Type”, “Res. Type”, “Skills” and “Phone”. An example “Java, DB Services” for the “Skills” or “987654321” for the “Phone”. The value of the XXX is the Attribute ID of the “Contract Type”, “Res. Type”, “Skills” or the “Phone”. All these values can be extracted from the UI on the configuration environment:



For the sequence of the list attribute values itself, they can be sent in an array, to make possible the entry of all values at once. The values can be extracted from the UI on the configuration environment:



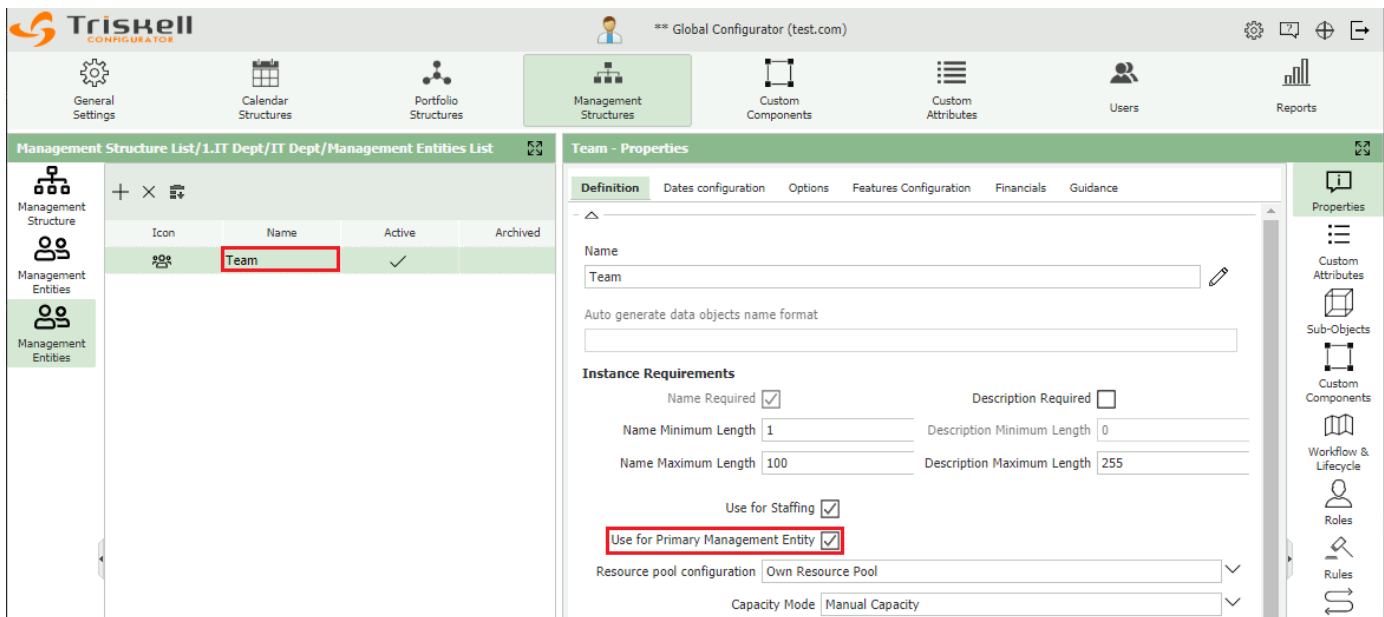
6 Auxiliary Services to get data

In this section it will be explained how to extract data required to call main services.

6.1 Get PME

In this section we describe how to get the data_object identifier from Triskell, identifier that to be used as a pme_id. Triskell allows extract data from your instance using reports. You need a stored selector to build your report.

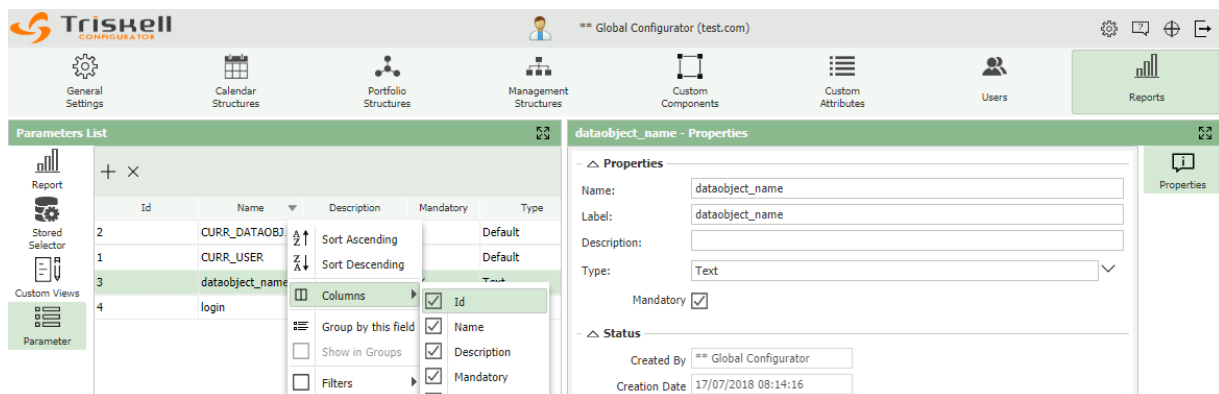
First, we need to get the object used for the “Primary Management Entity”:



The screenshot shows the Triskell Global Configurator interface. On the left, the 'Management Structure List/1.IT Dept/IT Dept/Management Entities List' is visible, with 'Team' selected. The main panel displays the 'Team - Properties' configuration. Under the 'Instance Requirements' section, the 'Use for Primary Management Entity' checkbox is checked and highlighted with a red box. Other settings include 'Name Required' (checked), 'Description Required' (unchecked), 'Name Minimum Length' (1), 'Name Maximum Length' (100), 'Description Minimum Length' (0), 'Description Maximum Length' (255), 'Use for Staffing' (checked), 'Resource pool configuration' (Own Resource Pool), and 'Capacity Mode' (Manual Capacity).

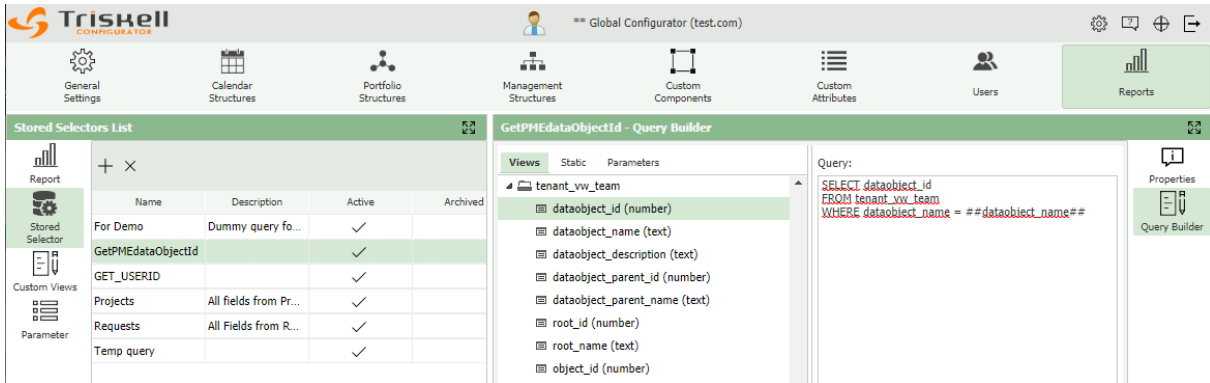
This report requires one parameter to get the data object id.

dataobject_name (Text):



The screenshot shows the Triskell Global Configurator interface. On the left, the 'Parameters List' is visible, with 'dataobject_name' selected. The main panel displays the 'dataobject_name - Properties' configuration. Under the 'Properties' section, the 'Name' is 'dataobject_name', the 'Label' is 'dataobject_name', the 'Type' is 'Text', and the 'Mandatory' checkbox is checked. Under the 'Status' section, the 'Created By' is '** Global Configurator' and the 'Creation Date' is '17/07/2018 08:14:16'.

Create the stored selector using the parameter:



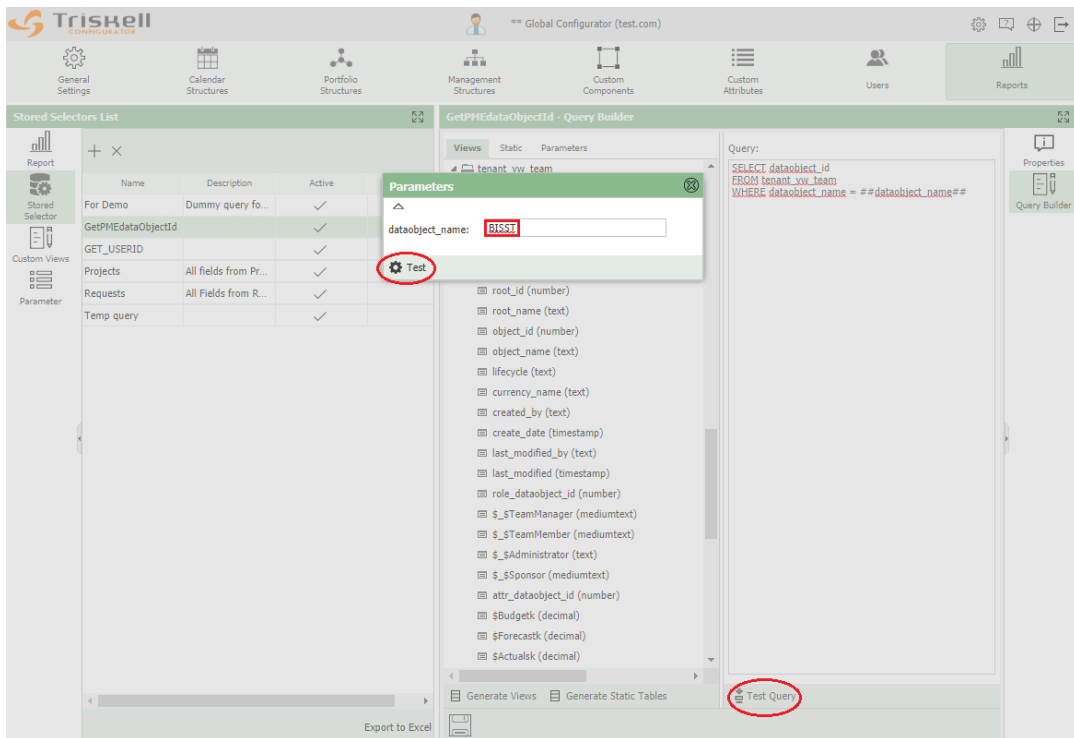
The screenshot shows the Triskell Global Configurator interface. On the left, the 'Stored Selectors List' table is visible:

Name	Description	Active	Archived
For Demo	Dummy query fo...	✓	
GetPMEdataObjectId		✓	
GET_USERID		✓	
Projects	All fields from Pr...	✓	
Requests	All Fields from R...	✓	
Temp query		✓	

The 'Query Builder' for 'GetPMEdataObjectId' is shown on the right. The 'Parameters' tab is active, showing a list of fields including 'dataobject_id (number)', 'dataobject_name (text)', 'dataobject_description (text)', 'dataobject_parent_id (number)', 'dataobject_parent_name (text)', 'root_id (number)', 'root_name (text)', and 'object_id (number)'. The 'Query' text area contains the following SQL query:

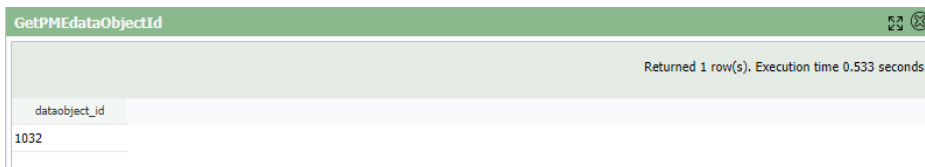
```
SELECT dataobject_id
FROM tenant_vw_team
WHERE dataobject_name = ##dataobject_name##
```

Then you can test the query to check that you get the id properly:



The screenshot shows the 'Parameters' dialog box for 'dataobject_name' with the value 'BISST' entered. The 'Test' button is highlighted with a red circle. The 'Test Query' button at the bottom right is also highlighted with a red circle. The SQL query in the background is the same as in the previous screenshot.

This is the result on this example:

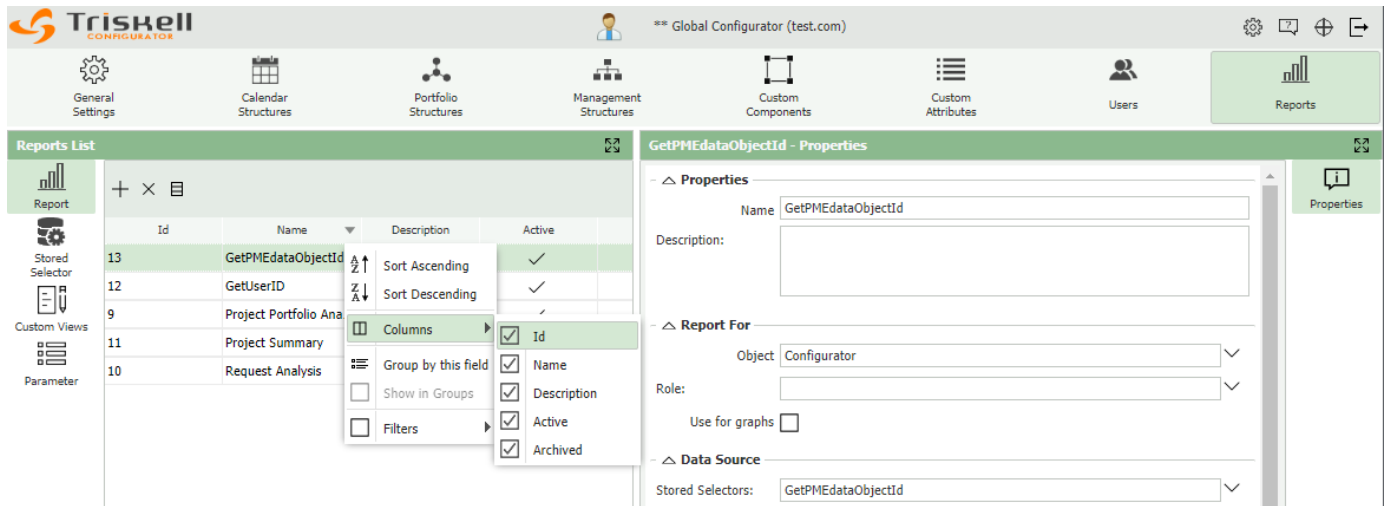


The screenshot shows the query result for 'GetPMEdataObjectId'. It indicates 'Returned 1 row(s). Execution time 0.533 seconds.' The result table has one column, 'dataobject_id', with the value '1032'.

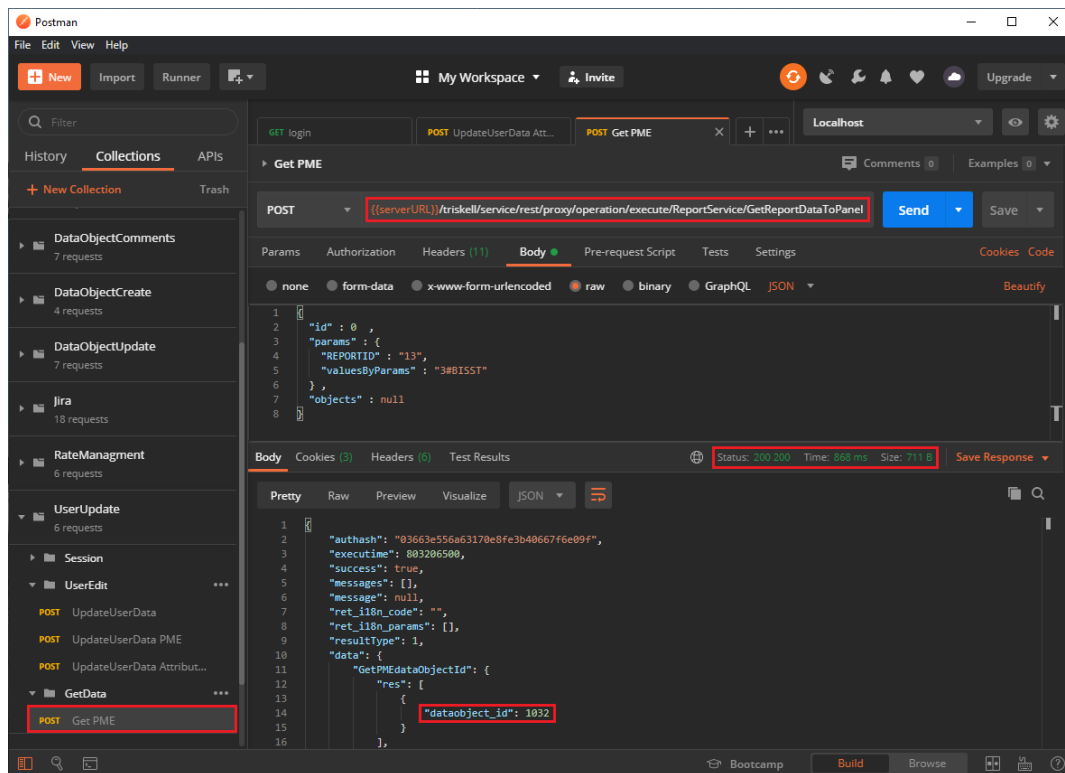
This is the query for the example above:

```
SELECT dataobject_id
FROM tenant_vw_team
WHERE dataobject_name = ##dataobject_name##
```

Once you create and test your stored selector, you need to create the associated report:



Now we have all id's needed and report to get period id dynamically from our API rest.



This is the example content request:

```

{
  "id": 0,
  "params": {
    "REPORTID": "13",
    "valuesByParams": "3#BISST"
  },
  "objects": null
}

```



Customer can do more than one request on the same REST conversation depending on your functional needs. This feature should not be used to load big data volumes because. During the REST conversation Triskell users can get blocked processing the request, so we suggest using it with common sense in terms of data volume and scheduling. Triskell will limit the number of requests by day in the future to avoid collapsing the server, so please take this in account when developing your interfaces.