REST API
Technical Documentation

Version 6.9.1

Get the solution you deserve

Contents

1 Document Change Management

Release	Date	Author	Comments
1.0	09/2014	Jose Manuel Prieto	First release
2.0	10/2015	Jose Ramón del Barrio	New Triskell Template New API REST services: • Tasks • Timesheet
3.0	02/2016	Jose Ramón del Barrio	New API REST service: • Timephased
4.0	05/2016	Jose Manuel Prieto	VBA examples with proxy requests
4.0	05/2016	Jose Ramón del Barrio	Allow timephased with comments
4.0	05/2016	Luis Figueira	Add currency to dataobjects rest service
4.1	04/2017	Jose Manuel Prieto	Add advanced lifecyle operational service
6.0	06/2017	Jose Ramón del Barrio	• Remove imposed start date from gantt. • New view to extract attributes id's tenant_vw_attributes_list.
6.1	02/2018	Jose Ramón del Barrio	Simpler way to extract data (GetData from ReportService)
6.2	11/2018	Jose Ramón del Barrio	Add additional comments to timephased
6.3	11/2019	Ricardo Fernandes	Create dataobjects with Auto generate name format
6.4	06/2020	Ricardo Fernandes	Add possibility to insert timesheets by user login
6.5	08/2020	Ricardo Fernandes	New data in the return of the create List Value service
6.6	09/2020	Ricardo Fernandes	New service to change List Values
6.7	10/2020	Ricardo Fernandes	New field and options in the user creation service
6.8	12/2020	Ricardo Fernandes	New timesheet fields in the user creation service
6.9	08/2021	Ricardo Fernandes	Add the creation on Gantt feature on Create Objects
6.9.1	03/2022	Ricardo Fernandes	Start/End dates on the creation of Gantt feature on Create Objects

2 Introduction

Several REST web services are provided allowing tenant and customers to interact with Triskell virtually from any environment/platform.

They are divided in two sections:

- **Data import services**: designed to help in the task of massive data import into the Triskell environment, mainly for initial and incremental data loads.
- **Service Proxies**: allowing invoking Triskell Operation or CRUD Services.

Client implementations for JAVA and/or VBA are provided for these services.

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.

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.

There are 3 distinct implementations : GET, POST and JSON.

3.1.1 GET

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' { pass } : String containing a Hex encoded MD5 hash of the password

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

3.1.2 POST

URL	https:// SERVER /triskell/service/rest/login
HTTP Method	POST
FORM Parameters	user : String containing user identifier as 'user@tenant.com' password : String containing a Hex encoded MD5 hash of the password

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

3.1.3 JSON

URL	https:// SERVER /triskell/service/rest/loginJson
HTTP Method	POST
JSON Parameters	user : String containing user identifier as 'user@tenant.com' password : String containing a Hex encoded MD5 hash of the password

```
{
        "user":"user@tenant.com",
        "password":"1234567890ABCDEF"
}
```

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

3.2 Logout Service

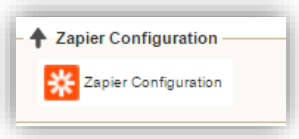
Logout service provided for closing the session at the server.

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

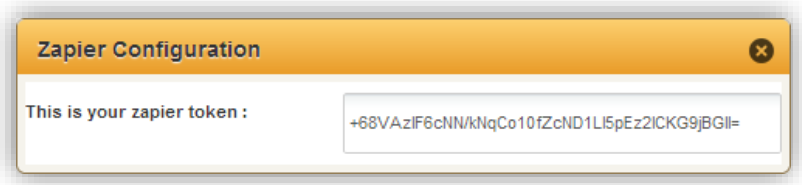
3.3 API Key

The API Key is the authentication token to be sent with the stateless services invocations.

It can be obtained through Triskell by going to the User Preferences Panel, and then clicking on the button:



<u>It opens a new window that gives the token for the authentication Header:</u>



4 Data Import

These services are designed to import large amount of data import into a Triskell tenant in a reliable and performing way.

Requests and responses are exchanged as XML documents with the clients, to enforce data correctness and validation.

These services can only be called by a Configurator user.
Any call from an unprivileged user will be rejected.

4.1 Dataobject Structure

This Web service is capable of searching dataobjects and importing complex dataobject structures into Triskell.

4.1.1 WS Dataobject Operation: Create

This WebService imports dataobjects into Triskell allowing adding to them:

- Relationships with other dataobjects
- Assign roles to users
- Custom Attribute values
- Nested dataobject children

The name of the dataobject must be provided (mandatory) by the user if the object of the dataobject to create don't have a "Auto generate data object name format". In this case the name must be provided in the field NAME of the XML.
On the opposite, if the object of the dataobject to create have a "generated code", the name it's not necessary (not mandatory), and can be empty in the field NAME of the XML. If the name is given to the dataobject, this name provided will be ignored and a new name will be generated by Triskell, just like in the configuration of the object.
The name generated by Triskell will be returned in the response XML, in the field "name_generated".
The dataobjectid will be also returned in the response XML, in the field "dataobjectid".

If the object of the dataobject to create on his own configuration have marked as true the option "*Use in Gantt?*", then at the time of the creation of the dataobject, it is also created the dataobject on the Gantt of the indicated parent dataobject. This feature it is not optional, to create data consistence on the system.
Another feature when creating the dataobject on the Gantt it is the possibility to send (or not) the Start/End date of the Item on the Gantt. To do that it is needed to send on the Custtom Attribute values, the dates that are configured on the object to be used as "*Gantt Chart Dates Set*". If they are provided, those dates are the ones used on the creation of the Item on the Gantt, if not, the Start date will be the current date and the End date the value of the current date plus the default duration value.

Every top level element and his nested children will be processed in a single transaction.

Any failure at child level will rollback all changes on the top level element.
After a top level element being processed successfully, failures at subsequent top level elements will not affect its status.

URL	https:// SERVER /triskell/service/rest/dataobject/create
HTTP Method	POST

Here is a sample of the input XML doc containing a list of dataobjects with examples on relationships, roles, children and custom attributes:

```
<?xml version="1.0" encoding="UTF-8"?>
<DATAOBJECTS>
  <DATAOBJECT>
    <OBJECT>Project</OBJECT>
    <NAME></NAME> <!-- This Object have a Auto generate name format, so the name can be empty -->
    <DESCRIPTION>Implement ISO 14 000 (evaluation year1 Implementation Y2-3)</DESCRIPTION>
    <CURRENCY>5</CURRENCY>
    <STAGE>1.Initiation</STAGE>
    <POOL>OWN</POOL>
    <PARENTID>1025</PARENTID> <!-- ONLY AT TOP LEVEL ELEMENTS -->

    <ATTRIBUTES>
      <ATTRIBUTE>
        <NAME>Start Date</NAME>
        <VALUE>2018.10.01</VALUE>
      </ATTRIBUTE>
      <ATTRIBUTE>
        <NAME>Finish date</NAME>
        <VALUE>2019.03.31</VALUE>
      </ATTRIBUTE>
      <ATTRIBUTE>
        <NAME>Exp. Benefits</NAME>
        <VALUE>10000</VALUE>
      </ATTRIBUTE>
      <ATTRIBUTE>
        <NAME>Risk</NAME>
        <VALUE>2.In Progress</VALUE>
      </ATTRIBUTE>
      <ATTRIBUTE>
        <NAME>Region</NAME>
        <VALUE>3.Japan</VALUE>
      </ATTRIBUTE>
      <ATTRIBUTE>
        <NAME>Goals</NAME>
        <VALUE>To create a data object via rest operation</VALUE>
      </ATTRIBUTE>
    </ATTRIBUTES>

    <RELATIONSHIPS>
        <RELATIONSHIP>
                <TYPE>Client</TYPE>
                <DATAOBJECTID>1024</DATAOBJECTID>
            </RELATIONSHIP>
    </RELATIONSHIPS>

    <USER_ROLES>
      <USER_ROLE>
        <ROLE>Project Manager</ROLE>
        <USER_CODE>Silva</USER_CODE>
      </USER_ROLE>
    </USER_ROLES>

    <CHILDS>
            <DATAOBJECT>
                <OBJECT>Work Package</OBJECT>
                <NAME>Package 001</NAME>
                <STAGE>1.Active</STAGE>
                <POOL>parent</POOL>
        <CURRENCY>5</CURRENCY>
```

```
                    <RELATIONSHIPS></RELATIONSHIPS>
                    <USER_ROLES></USER_ROLES>
                    <ATTRIBUTES></ATTRIBUTES>

                    <CHILDS>
                        <DATAOBJECT>
                            <OBJECT>Task</OBJECT>
                            <NAME>Task 1</NAME>
                            <STAGE>- NA -</STAGE>
                            <POOL>parent</POOL>
                <CURRENCY>5</CURRENCY>

                            <RELATIONSHIPS> ... </RELATIONSHIPS>
                            <USER_ROLES> ... </USER_ROLES>
                            <ATTRIBUTES> ... </ATTRIBUTES>

                        </DATAOBJECT>

                        <DATAOBJECT>
                            <OBJECT>Task</OBJECT>
                            <NAME>Task 2</NAME>
                            <STAGE>- NA -</STAGE>
                            <POOL>parent</POOL>
                <CURRENCY>5</CURRENCY>

                            <RELATIONSHIPS> ... </RELATIONSHIPS>
                            <USER_ROLES> ... </USER_ROLES>
                            <ATTRIBUTES> ... </ATTRIBUTES>
                        </DATAOBJECT>

                        <DATAOBJECT> ... </DATAOBJECT>
                        <DATAOBJECT> ... </DATAOBJECT>
                    </CHILDS>
                </DATAOBJECT>
            </CHILDS>
    </DATAOBJECT>

        <DATAOBJECT> ... </DATAOBJECT>
        <DATAOBJECT> ... </DATAOBJECT>
</DATAOBJECTS>
```

It returns a XML with a list of results codes OK/KO for every dataobject into the request, including an error description on failures.

```
<?xml version='1.0' encoding='UTF-8'?>
<dataobjects>
  <dataobject>
    <name>Task 1</name>
    <result>OK</result>
    <dataobjectid>2097</dataobjectid>
    <name_generated></name_generated>
  </dataobject>
  <dataobject>
    <name>Task 2</name>
    <result>OK</result>
    <dataobjectid>2098</dataobjectid>
    <name_generated></name_generated>
  </dataobject>
  <dataobject>
    <name>Package 001</name>
```

```
      <result>OK</result>
      <dataobjectid>2096</dataobjectid>
      <name_generated></name_generated>
   </dataobject>
   <dataobject>
      <name></name>
      <result>OK</result>
      <dataobjectid>2095</dataobjectid>
      <name_generated>PRJ 046</name_generated>
   </dataobject>
</dataobjects>
```

4.1.2 WS Dataobject Operation: Search

This operation searches for dataobjects having their URL parameter {at} equals to the provided String at the URL parameter {toSearch}, with the exception of the option 'CHILDS' that returns all dataobjects having as parent the dataobject identified by id at {toSearch}.

URL	https:// SERVER /triskell/service/rest/dataobject/search/{toSearch}/at/{at}
HTTP Method	GET
URL Parameters	{at}: • NAME : search dataobjects by name • DESCRIPTION : search dataobjects by description • CHILDS : Obtain all dataobject children by his identifier

It returns a XML document containing dataobject nodes with his name and identifier.

4.1.3 Java Client

The Triskell Dataobject WS client for JAVA will be provided in an Eclipse project containing all the needed classes and dependencies.

Three client versions are provided for working with: XML Documents as String, W3C documents and Xom documents.

Package	eu.triskell.ws.rest.client.businessDelegate.dataObject
Class	DataObjectBD<E>
Direct Known Subclasses	DataObjectBDString DataObjectBDW3CDom DataObjectBDXom

Constructor Summary	
protected	**DataObjectBD(Conversation c, java.lang.Class<E> clazz)** Constructor with a Conversation object that will manage the state against the server.

Method Summary	
E	**create(E doc)** Receives a xml like : <DATAOBJECTS> <DATAOBJECT>...
E	**searchAt(java.lang.String search, java.lang.String at)** Search any.
E	**searchByDescription(java.lang.String desc)** Search Dataobjects by Description.
E	**searchByName(java.lang.String name)** Search Dataobjects by Name.
E	**searchChildsOf(java.lang.Long id)** Search Dataobjects nesting the given dataobjectId.

This is an example on how to call the create operation with the JAVA client:

```
Conversation conversation = null;
```

```
try {
        conversation = new Conversation("https://ondemand.triskellsoftware.com");

        conversation.logOn("admin@tenant.com", "aeiou123");
} catch (TkRestClientException e) {
        e.printStackTrace();
        return;
}

DataObjectBDW3CDom dobd = new DataObjectBDW3CDom(conversation);

org.w3c.dom.Document    request = buildRequestDocument(),
                        response;

response = dobd.create(request);
```

4.2 Users

This Web Service allows to create and to search Triskell lists of user/login.

4.2.1 WS Users Operation: Create

This operation creates users/logins into a Triskell tenant.

URL	https:// SERVER /triskell/service/rest/user/create
HTTP Method	POST

It accepts a list of users into an XML with the following structure:

```
<?xml version="1.0">
<USERS>
        <USER>
                <CODE>tk123</CODE>
                <NAME>Foo Bar</NAME>
                <EMAIL>foo.bar@barfoo.eu</EMAIL>

                <MD5PASSWORD>b4ee0c0038802d2903369d530b13c7d3</MD5PASSWORD>
                OR
                <PASSWORD>AEIOU123</PASSWORD>

                <LANGUAGECODE>EN</LANGUAGECODE>
                OR
                <LANGUAGE>English</LANGUAGE>

                <CALENDAR>Company Calendar</CALENDAR>
                <ACTIVE>(on / true / 1)</ACTIVE>
                <ARCHIVED>(off / false / 0)</ARCHIVED>
                <CONFIGURATOR>(off / false / 0)</CONFIGURATOR>
                <DEFAULT_STARTPAGE>Timesheet Page</DEFAULT_STARTPAGE>

                <TIMESHEET>true</TIMESHEET>
                <CANENTERONNONWORKINGDAYS>true</CANENTERONNONWORKINGDAYS>
                <ALLOWNPAS>true</ALLOWNPAS>
                <MINVALUE>1</MINVALUE>
                <MAXVALUE>8</MAXVALUE>
                 <DEFAULTVALUE>7</DEFAULTVALUE>

                <PME>
                        <NAME>BSIC</NAME>
                        <START_DATE>2020.10.01</START_DATE>
                </PME>
        </USER>
        <USER>
                <CODE>tk321</CODE>
                ...
        </USER>
        ...
</USERS>
```

It returns an XML with a list of results codes OK/KO for every user into the request, including an error description on failures:

```
<?xml version="1.0"?>
<users>
```

```
        <user>
                <code>tk123</code>
                <result>OK</result>
                <identifier>287</identifier>
        </user>
        <user>
                <code>tk321</code>
                <result>KO</result>
                <error>description</error>
        </user>
<users>
```

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.

4.2.1.1 Fields description

4.2.1.1.1 Mandatory fields

- code: unique user identifier used as a login (String).
- name: the name to the user (String).
- password: the password of the user to make login (String).
- md5password: the password of the user in MD5 format to make login (String).
- languageCode: unique identifier of the language to be used by the user (String).
- calendar: unique identifier of the default calendar of the user (String).

4.2.1.1.2 Optional fields

- email: the email of the user (String).
- active: Boolean value (true/false).
- archived: Boolean value (true/false).
- configurator: Boolean value (true/false).
- default_startPage: unique identifier of the default start page to the user (String). Have fixed and dynamic values.
    - Fixed:
        - Application Home Page
        - Management Structures Page
        - Portfolios Page
        - Portfolio Items Page
        - Custom Components Page
        - Management Entities Page
        - Timesheet Page
        - Calendar Page
        - Reports Page
    - Dynamic:
        - The name of the Object available with the option “Show Opt in Main Menu” with value YES on is definition (Ex: Project, Milestone…)
- timesheet: 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.
- maxValue: Big Decimal value with decimal separator "." and no thousands separator.
- defaultValue: Big Decimal value with decimal separator "." and no thousands separator.
- PME: group of fields to add the Primary Management Entity of the user.
    - name: unique name identifier of the DataObject to use as a PME.
    - start_date: unique Date in the day format (String). The Date must be in the format "YYYY.MM.DD" to work, without hours, minutes and seconds.

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

4.2.1.2 User Creation example

There are some examples available about how to send information to Triskell using Postman Google plug-in. You can get postman free here. Then you just need to import the attached file "*UserCreation.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.

4.2.2 WS Users Operation: Search

This operation search for users having active the flag provided at the URL parameter {what}.

URL	https:// SERVER /triskell/service/rest/user/search/{what}
HTTP Method	GET
URL Parameters	• ALL : All tenant Users • ACTIVE : All active Users • ARCHIVED : All archived Users Example for searching all users: HTTPS:// SERVER /triskell/service/rest/user/search/ALL

It returns a XML document with a list of users matching the condition:

```
<?xml version="1.0">
<USERS>
        <USER>
                <CODE>tk123</CODE>
                <NAME>Foo Bar</NAME>
                <EMAIL>foo.bar@barfoo.eu</EMAIL>
                <LANGUAGECODE>EN</LANGUAGECODE>
                <CALENDAR>Company Calendar</CALENDAR>
                <ACTIVE>(on / true / 1)</ACTIVE>
                <ARCHIVED>(off / false / 0)</ARCHIVED>
                <TIMESHEET>(on / true / 1)</TIMESHEET>
                <CONFIGURATOR>(off / false / 0)</CONFIGURATOR>
        </USER>
        ...
</USERS>
```

4.2.3 WS Users Operation : Search At

This operation search for users having his URL parameter {where} equals to the provided String at the URL parameter {toSearch}.

URL	https:// SERVER /triskell/service/rest/user/search/{toSearch}/at/{where}
HTTP Method	GET
URL Parameters	{where}: • NAME : User Name/Description • CODE : User Code/Login (without tenant domain) Example for searching a user called John Smith: https:// SERVER /triskell/service/rest/user/search/John%20Smith/at/NAME

<u>It returns a XML document with a list of users matching the condition:</u>

```
<?xml version="1.0">
<USERS>
        <USER>
```

```
                <CODE>tk123</CODE>
                <NAME>Foo Bar</NAME>
                <EMAIL>foo.bar@barfoo.eu</EMAIL>
                <LANGUAGECODE>EN</LANGUAGECODE>
                <CALENDAR>Company Calendar</CALENDAR>
                <ACTIVE>(on / true / 1)</ACTIVE>
                <ARCHIVED>(off / false / 0)</ARCHIVED>
                <TIMESHEET>(on / true / 1)</TIMESHEET>
                <CONFIGURATOR>(off / false / 0)</CONFIGURATOR>
        </USER>
        ...
</USERS>
```

4.2.4 WS Users Java Client

The Triskell User WS client for JAVA will be provided into an Eclipse project containing all the needed classes and dependencies.

Three client versions are provided for working with: XML Documents as String, W3C documents and Xom documents.

Package	eu.triskell.ws.rest.client.businessDelegate.user
Class	UserBD<E>
Direct Known Subclasses	UserBDString UserBDW3CDom UserBDXom

<u>Business Delegate for accessing the Triskell User Rest Services:</u>

Constructor Summary	
protected	UserBD(Conversation c, java.lang.Class<E> clazz) Constructor with a Conversation object that will manage the state against the server.

Method Summary	
E	**create(E doc)** Receives a xml
E	**getActive()** Returns all the tenant Active users.
E	**getAll()** Returns all the tenant users.
E	**getArchived()** Returns all the tenant Archived users.
E	**searchAt(java.lang.String search, java.lang.String at)** Search any.
E	**searchByCode(java.lang.String code)** Search a user by code.
E	**searchByName(java.lang.String name)**

Search a user by name.

This is an example on how to call the create operation with the JAVA client:

```
Conversation conversation = null;

try {
	conversation = new Conversation("https://ondemand.triskellsoftware.com");

	conversation.logOn("admin@tenant.com", "aeiou123");
} catch (TkRestClientException e) {
	e.printStackTrace();
	return;
}

UserBDW3CDom usbd = new UserBDW3CDom (conversation);

org.w3c.dom.Document	request = buildRequestDocument(), response;

response = usbd.create(request);
```

4.3 Custom Attribute List Values

This web service allows adding values into an existing Custom Attribute of type List.

4.3.1 WS AttrListVal Operation: Create

This operation adds new values into an existing 'Custom Attribute List' into a Triskell tenant. These values can be all at the same level or in a future to a nested on a tree configuration.

URL	https:// SERVER /triskell/service/rest/cutomAttributeListValues/create/at/{attrId}
HTTP Method	POST
URL Parameters	{ attrId } is the a numerical identifier of a custom attribute of type list

It accepts a list of values for an existing 'Custom Attribute List' into an XML with the following structure:

```
<?xml version="1.0"?>
<CustomAttrListValues>
  <CustomAttrListValue>
    <active> true/false </active>
    <archived> true/false </archived>
    <defaultvalue> true/false </defaultvalue>
    <selectable> true/false </selectable>
    <scoreVal> 1 </scoreVal>
    <label> The Label </label>
    <comment> A comment </comment>
    <backgroundColor> RGB #FFFFFF </backgroundColor>
    <fontColor> RGB #FFFFFF </fontColor>
    <previewImageExtension> The image extension : PNG, GIF, BMP ... </previewImageExtension>
    <imageByteBuffer> Base64 encoded image byte array </imageByteBuffer>

    <!-- this Tree feature will be implemented in the future -->
    <!--
    <childs>
      <CustomAttrListValue>
        <active> true/false </active>
        <archived> true/false </archived>
        <defaultvalue> true/false </defaultvalue>
        <selectable> true/false </selectable>
        <scoreVal> 2 </scoreVal>
        <label> The Label </label>
        <comment> A comment </comment>
        <backgroundColor> RGB #FFFFFF </backgroundColor>
        <fontColor> RGB #FFFFFF </fontColor>
        <previewImageExtension> extension : PNG, GIF, BMP ... </previewImageExtension>
        <imageByteBuffer> Base64 encoded image byte array </imageByteBuffer>
      </CustomAttrListValue>
    </childs>
    <-->
  </CustomAttrListValue>
</CustomAttrListValues>
```

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.

The answer of this request will return some data of the Custom Attribute Value List created. The user can do what they want with that information. The answer is one XML with the structure in an Array of the values created.

This is the fields returned by the answer:

- label
- sequence
- customattrlistvalueid
- result

This is one example of the answer:

```
<?xml version="1.0"?>
<customattrlistvalues>
  <customattrlistvalue>
    <label>AAA</label>
    <result>OK</result>
    <customattrlistvalueid>289</customattrlistvalueid>
    <sequence>1</sequence>
  </customattrlistvalue>
  <customattrlistvalue>
    <label>BBB</label>
    <result>OK</result>
    <customattrlistvalueid>290</customattrlistvalueid>
    <sequence>2</sequence>
  </customattrlistvalue>
</customattrlistvalues>
```

4.3.1.1 Fields description

4.3.1.1.1 Mandatory fields

- attrId: unique Custom Attribute identifier (Integer). This must be an ID of one Custom Attribute **List**.
- label: the value for the label to the Custom Attribute List Value (String).

4.3.1.1.2 Optional fields

- active: Boolean value (true/false).
- archived: Boolean value (true/false).
- defaultvalue: Boolean value (true/false).
- selectable: Boolean value (true/false).
- scoreval: Integer value to be the Score.
- comment: String value.
- backgroundColor: String value in the #RGB format.
- fontColor: String value in the #RGB format.
- previewImageExtension: String value.
- imageByteBuffer: String value with the file encoded in Base64.

4.3.2 WS AttrListVal Operation: update

This operation allows update values in an existing 'Custom Attribute List' into a Triskell tenant. These values must be all at the same level or in a future to a nested on a tree configuration.

URL	https:// SERVER /triskell/service/rest/cutomAttributeListValues/update/at/{attrId}
HTTP Method	POST
URL Parameters	{ attrId } is the a numerical identifier of a custom attribute of type list

It accepts a list of values for an existing 'Custom Attribute List Values' into an XML with the following structure:

```
<?xml version="1.0"?>
<CustomAttrListValues>
    <CustomAttrListValue>
      <customattrlistvalueid>252</customattrlistvalueid>
      <label>AAA</label>

      <active>true</active>
      <archived>false</archived>
      <defaultvalue>true</defaultvalue>
      <selectable>false</selectable>
      <scoreVal>1</scoreVal>

      <comment>Now Have this comment AAA</comment>

      <!--
      <backgroundColor> RGB #FFFFFF </backgroundColor>
      <fontColor> RGB #FFFFFF </fontColor>
      <previewImageExtension> The image extension : PNG, GIF, BMP ... </previewImageExtension>
      <imageByteBuffer> Base64 encoded image byte array </imageByteBuffer>
      -->
   </CustomAttrListValue>

   <CustomAttrListValue>
      <!-- Mandatory fields -->
      <customattrlistvalueid>253</customattrlistvalueid>
      <label>BBB</label>

      <!-- Fields with default value -->
      <active>false</active>
      <archived>true</archived>
      <defaultvalue>false</defaultvalue>
      <selectable>true</selectable>
      <scoreVal>2</scoreVal>

      <backgroundColor>#CF2424</backgroundColor>
      <fontColor>#2E8F0C</fontColor>

      <!-- Optional fields -->
      <comment>Now Have this comment BBB</comment>

      <previewImageExtension>png</previewImageExtension>
      <imageByteBuffer> Base64 encoded image byte array </imageByteBuffer>
   </CustomAttrListValue>
</CustomAttrListValues>
```

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/update 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.

The answer of this request will return some data of the Custom Attribute Value List created. The user can do what they want with that information. The answer is one XML with the structure in an Array of the values created.

This is the fields returned by the answer:

- label
- sequence
- customattrlistvalueid
- result

This is one example of the answer:

```
<?xml version="1.0"?>
<customattrlistvalues>
  <customattrlistvalue>
    <label>AAA</label>
    <result>OK</result>
    <customattrlistvalueid>289</customattrlistvalueid>
    <sequence>1</sequence>
  </customattrlistvalue>
  <customattrlistvalue>
    <label>BBB</label>
    <result>OK</result>
    <customattrlistvalueid>290</customattrlistvalueid>
    <sequence>2</sequence>
  </customattrlistvalue>
</customattrlistvalues>
```

4.3.2.1 Fields description

4.3.2.1.1 Mandatory fields

- attrId: unique Custom Attribute identifier (Integer). This must be an ID of one Custom Attribute **List**.
- customattrlistvalueid: unique Custom Attribute List Value identifier (integer). This must be an ID of one value from the **previous attrId**.
- label: the value for the label to the Custom Attribute List Value (String).

4.3.2.1.2 Optional fields

- active: Boolean value (true/false).
- archived: Boolean value (true/false).
- defaultvalue: Boolean value (true/false).
- selectable: Boolean value (true/false).
- scoreval: Integer value to be the Score.
- comment: String value.
- backgroundColor: String value in the #RGB format.
- fontColor: String value in the #RGB format.
- previewImageExtension: String value.
- imageByteBuffer: String value with the file encoded in Base64.

Some of this optional fields have a default value, and if the value is not sent in the update call, it will be updated to the **default** value. See the comments in the XML example above.

These are fields with default values and the respect default value:

- active: true
- archived: false
- defaultvalue: false
- selectable: true
- scoreval: 0
- backgroundColor: #FFFFFF (white)
- fontColor: #000000 (black)

4.3.3 WS AttrListVal Java Client

The Triskell CustomAttributeListValues WS client for JAVA will be provided into an Eclipse project containing all the needed classes and dependencies.

Three client versions are provided for working with: XML Documents as String, W3C documents and Xom documents.

Package	eu.triskell.ws.rest.client.businessDelegate.customAttributeListValues
Class	CustomAttributeListValuesBD<E>
Direct Known Subclasses	CustomAttributeListValuesBDString CustomAttributeListValuesBDW3CDom CustomAttributeListValuesBDXom

Constructor Summary	
protected	**CustomAttributeListValuesBD(Conversation c, java.lang.Class<E> clazz)** Constructor with a Conversation object that will manage the state against the server.

Method Summary	
E	**create(java.lang.Integer attrId, E doc)** Receives an xml like : <CustomAttrListValues>. <CustomAttrListValue> <active> true/false </active> <archived> true/false </archived> <defaultvalue> true/false </defaultvalue> <selectable> true/false </selectable> <scoreVal> 1 </scoreVal> <label> The Label </label> <comment> A comment </comment> <backgroundColor> RGB #FFFFFF </backgroundColor> <fontColor> RGB #FFFFFF </fontColor> <previewImageExtension> The image extension : PNG, GIF, BMP ...

<u>This is an example on how to call the create operation with the JAVA client:</u>

```
Conversation conversation = null;

try {
	conversation = new Conversation("https://ondemand.triskellsoftware.com");

	conversation.logOn("admin@tenant.com", "aeiou123");
} catch (TkRestClientException e) {
	e.printStackTrace();
	return;
}
```

```
CustomAttributeListValuesBDW3CDom cattrbd = new CustomAttributeListValuesBDW3CDom(conversation);

org.w3c.dom.Document            request = buildRequestDocument(),
                                response;

response = cattrbd.create(123, request); /* 123 is the custom attribute identifier */
```

4.4 Gantt Custom Component Create

This web service allows add Gantt custom components to an existing dataobject.

- It will run only if it's properly configured.
- It does not update existing Gantt custom components.
- It's only available if dataobject doesn't have Gantt custom component yet.
- It requires an initial login authentication and it's only accessible for configurator users.

This service is different from standard dataobjects creation because its behavior is completely specific. So please do not use dataobjects web service creation to create Gantt custom components.

URL	URL: https:// SERVER /triskell/service/rest/tasks/create
HTTP Method	POST

It accepts an xml with the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="https://ondemand.triskellsoftware.com/xsd/tasks"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="https://ondemand.triskellsoftware.com/xsd/tasks">

        <xsd:annotation>
                <xsd:documentation>Xml definition Tasks RESTful WebService
                </xsd:documentation>
        </xsd:annotation>

        <xsd:element name="dataobjects" type="dataobjectsType" />

        <xsd:complexType name="attributeType">
                <xsd:sequence>
                        <xsd:element type="xsd:string" name="name" />
                        <xsd:element type="xsd:string" name="value" />
                </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="attributesType">
                <xsd:sequence>
                        <xsd:element type="attributeType" name="attribute"
                                maxOccurs="unbounded" minOccurs="0" />
                </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="dataobjectType">
                <xsd:sequence>
                        <xsd:element type="xsd:string" name="taskid" />
                        <xsd:element type="xsd:string" name="name" />
                        <xsd:element type="xsd:date" name="StartDate" />
                        <xsd:element type="xsd:date" name="EndDate" />
                        <xsd:element type="xsd:int" name="Duration" />
                        <xsd:element type="xsd:string" name="DurationUnit" />
                        <xsd:element type="xsd:string" name="SchedulingMode" />
                        <xsd:element type="xsd:date" name="actualStart" minOccurs="0" />
                        <xsd:element type="xsd:date" name="actualEnd" minOccurs="0" />
                        <xsd:element type="xsd:float" name="Effort" />
                        <xsd:element type="xsd:string" name="EffortUnit" />
                        <xsd:element type="xsd:decimal" name="PercentDone" />
                        <xsd:element type="attributesType" name="attributes" minOccurs="0"  />
                        <xsd:element type="childTasksType" name="childTasks" minOccurs="0"  />
```

```
            </sxd:sequence>
        </xsd:complexType>

        <xsd:complexType name="childTasksType">
            <xsd:sequence>
                <xsd:element type="dataobjectType" name="dataobject"
                    maxOccurs="unbounded" minOccurs="0" />
            </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="dataobjectsType">
            <xsd:sequence>
                <xsd:element type="xsd:string" name="object" />
                <xsd:element type="xsd:string" name="name" />
                <xsd:element type="childTasksType" name="childTasks" />
            </xsd:sequence>
        </xsd:complexType>

</xsd:schema>
```

This is a XML example:

```
<?xml version='1.0' encoding='UTF-8'?>
<p:dataobjects xmlns:p="https://ondemand.triskellsoftware.com/xsd/tasks"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="https://ondemand.triskellsoftware.com/xsd/tasks tasks.xsd">
        <object>Project</object>
        <name>TEST</name>
        <childTasks>
            <dataobject>
                <taskid>2</taskid>
                <name>Task 1</name>
                <StartDate>2015-09-10</StartDate>
                <EndDate>2015-09-10</EndDate>
                <Duration>5</Duration>
                <DurationUnit>h</DurationUnit>
                <SchedulingMode>FixedDuration</SchedulingMode>
                <Effort>0</Effort>
                <EffortUnit>h</EffortUnit>
                <PercentDone>0</PercentDone>
                <childTasks>
                    <dataobject>
                        <taskid>3</taskid>
                        <name>Task 2</name>
                        <StartDate>2015-09-10</StartDate>
                        <EndDate>2015-09-10</EndDate>
                        <Duration>5</Duration>
                        <DurationUnit>h</DurationUnit>
                        <SchedulingMode>FixedDuration</SchedulingMode>
                        <Effort>0</Effort>
                        <EffortUnit>h</EffortUnit>
                        <PercentDone>0</PercentDone>
                        <attributes>
                            <attribute>
                                <name>Attribute name</name>
                                <value>Value</value>
                            </attribute>
                        </attributes>
                    </dataobject>
```

```
                                <dataobject>
                                        <taskid>4</taskid>
                                        <name>Task 2</name>
                                        <StartDate>2015-09-10</StartDate>
                                        <EndDate>2015-09-10</EndDate>
                                        <Duration>5</Duration>
                                        <DurationUnit>h</DurationUnit>
                                        <SchedulingMode>FixedDuration</SchedulingMode>
                                        <Effort>0</Effort>
                                        <EffortUnit>h</EffortUnit>
                                        <PercentDone>0</PercentDone>
                                        <attributes>
                                                <attribute>
                                                        <name>Attribute name</name>
                                                        <value>Value</value>
                                                </attribute>
                                        </attributes>
                                </dataobject>
                        </childTasks>
                </dataobject>
        </childTasks>
</p:dataobjects>
```

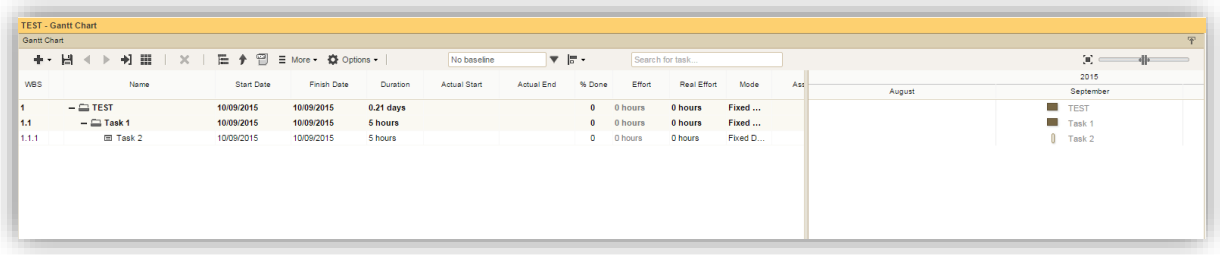
Here is the description for some of these fields:

- Taskid: is a string unique identifier for every task. It's not a Triskell identifier and it's required to manage the web service response. It won't be stored in Triskell.
- Date fields are in ISO format (YYYY-MM-DD).
- Units allowed values:
  - h:hours
  - d:days
- Scheduling Modes allowed values:
  - FixedDuration
  - EffortDriven
  - DynamicAssignment

This is a response example (XML):

```
<?xml version='1.0' encoding='UTF-8'?>
        <dataobjects>
                <data>
                        <result>OK</result>
                        <taskid>2</taskid>
                </data>
                <data>
                        <result>OK</result>
                        <taskid>3</taskid>
                </data>
</dataobjects>
```

If all results are ok and you access your Triskell environment you can see the data loaded:



Data will be "normalized" after Gantt reloading applying Triskell Gantt rules and calendar. It can produce some differences between the request and displayed data.

This web service version only allows load text Gantt custom component attributes.

No Java Client is provided to use this service but you can find a request example on Triskell-ImportTasks.xlsm.

4.5 Timesheet Create

This web service allows add/update timesheet data.

- It will only apply if it's properly configured.
- It requires an initial login authentication and it's only accessible for configurator users.
- It's only available to load daily data.

URL	URL: https:// SERVER /triskell/service/rest/timesheet/create
HTTP Method	POST

It accepts an xml with the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema		xmlns="https://ondemand.triskellsoftware.com/xsd/timesheet"
		xmlns:xsd="http://www.w3.org/2001/XMLSchema"
		targetNamespace="https://ondemand.triskellsoftware.com/xsd/timesheet">

	<xsd:annotation>
	 <xsd:documentation>Xml definition 4 Timesheet RESTful WebService</xsd:documentation>
	</xsd:annotation>

	<xsd:element name="timesheets" type="UserListType"/>

	<xsd:complexType name="UserListType">
		<xsd:sequence>
			<xsd:element name="user" type="UserType" minOccurs="1" maxOccurs="1" />
		</xsd:sequence>
	</xsd:complexType>

	<xsd:complexType name="UserType">
		<xsd:sequence>
			<xsd:element name="type" type="xsd:string" minOccurs="0" maxOccurs="1" />
			<xsd:element name="value" type="xsd:string" minOccurs="1" maxOccurs="1" />
			<xsd:element name="object" type="ObjectType" minOccurs="1" maxOccurs="unbounded" />
		</xsd:sequence>
	</xsd:complexType>

	<xsd:complexType name="ObjectType">
		<xsd:sequence>
			<xsd:element name="path" type="xsd:string" minOccurs="1" maxOccurs="1" />
			<xsd:element name="timesheet" type="DataListType" minOccurs="1" maxOccurs="1" />
		</xsd:sequence>
	</xsd:complexType>

	<xsd:complexType name="DataListType">
		<xsd:sequence>
			<xsd:element name="data" type="DataType" minOccurs="1" maxOccurs="unbounded" />
		</xsd:sequence>
	</xsd:complexType>

	<xsd:complexType name="AttributeType">
		<xsd:sequence>
			<xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="1" />
			<xsd:element name="value" type="xsd:string" minOccurs="1" maxOccurs="1" />
		</xsd:sequence>
	</xsd:complexType>

	<xsd:complexType name="AttributesListType">
		<xsd:sequence>
			<xsd:element name="attribute" type="AttributeType" maxOccurs="5" />
```

```
            </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="DataType">
            <xsd:sequence>
                <xsd:element name="timesheetid" type="xsd:string" minOccurs="1" maxOccurs="1" />
                <xsd:element name="attributes" type="AttributesListType" minOccurs="0" maxOccurs="1" />
                <xsd:element name="value" type="xsd:date" minOccurs="1" maxOccurs="1" />
                <xsd:element name="units" type="xsd:float" minOccurs="1" maxOccurs="1" />
                <xsd:element name="etc" type="xsd:float" minOccurs="1" maxOccurs="1" />
            </xsd:sequence>
        </xsd:complexType>

        <xsd:simpleType name="ValueType">
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="\s{0}[_A-Z0-9]+\s{0}" />
            </xsd:restriction>
        </xsd:simpleType>
</xsd:schema>
```

This is an XML example:

```
<p:timesheets xmlns:p="https://ondemand.triskellsoftware.com/xsd/timesheet"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://ondemand.triskellsoftware.com/xsd/timesheet timesheet-1.0.xsd ">
        <user>
            <type>name</type>
            <value>User 01</value>
            <object>
                <path>Global IT Portfolio/Project 01/Module 1/Task 1</path>
                <timesheet>
                    <data>
                        <timesheetid>7655eb25-e5c8-460d8f28cce</timesheetid>
                        <attributes>
                            <attribute>
                                <name>Billable?</name>
                                <value>No</value>
                            </attribute>
                        </attributes>
                        <value>2015-09-17</value>
                        <units>8.0</units>
                        <etc>0</etc>
                    </data>
                    <data>
                        <timesheetid>8655eb25-e5c8-460d8f28cce</timesheetid>
                        <attributes>
                            <attribute>
                                <name>Billable?</name>
                                <value>Yes</value>
                            </attribute>
                        </attributes>
                        <value>2015-09-18</value>
                        <units>8.0</units>
                        <etc>0</etc>
                    </data>
                    <data>
                        <timesheetid>9655eb25-e5c8-460d8f28cce</timesheetid>
                        <attributes>
                            <attribute>
```

```
                                                    <name>Billable?</name>
                                                    <value>No</value>
                                            </attribute>
                                    </attributes>
                                    <value>2015-09-16</value>
                                    <units>8.0</units>
                                    <etc>0</etc>
                            </data>
                    </timesheet>
            </object>
    </user>
</p:timesheets>
```

Here is the description for some of these fields:

- Type: The field name used to find/get the user (*optional*)*
  - Possible values: name, login
- Value: The name or the login (indicated in the Type) of the user to create/update the timesheet (*mandatory*)
- Path: Complete path for a custom component. Separator "/". (*mandatory*)
- timesheetid: is a unique string identifier for a timesheet request. (*mandatory*)
- Date field is ISO format (YYYY-MM-DD)

*Note: The Type is an optional field, so if it is **not added** to the call it will find/get the user by his **name**, just like in the previous version of this REST service. This is to be retro-compatible with the previous version and clients don't need to make any change on their implementations.

This is a response XML example:

```
<?xml version='1.0' encoding='UTF-8'?>
<timesheets>
  <data>
    <result>OK</result>
    <timesheetid>7655eb25-e5c8-460d8f28cce</timesheetid>
  </data>
  <data>
    <result>OK</result>
    <timesheetid>8655eb25-e5c8-460d8f28cce</timesheetid>
  </data>
  <data>
    <result>OK</result>
    <timesheetid>9655eb25-e5c8-460d8f28cce</timesheetid>
  </data>
</timesheets>
```

This web service only allows daily data

No Java Client is provided to use this service

4.6 Timephased Attribute Data

This web service allows add/update timephased data.

- It will only apply if it's properly configured.
- It requires an initial login authentication and it is accessible for all Triskell users and it will take it account same security rules.
- It's only available to load monthly data.

URL	URL: https:// SERVER /triskell/service/rest/timephased/update
HTTP Method	POST

It accepts an xml with the following schema (To be changed in the future):

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="timephased">
  <xs:complexType>
   <xs:sequence>
    <xs:element type="xs:int" name="object"/>
    <xs:element type="xs:int" name="timephased_attribute"/>
    <xs:element type="xs:int" name="rate_type"/>
    <xs:element type="xs:string" name="path"/>
    <xs:element type="xs:string" name="name"/>
    <xs:element name="version" maxOccurs="unbounded" minOccurs="0">
     <xs:complexType>
      <xs:sequence>
       <xs:element type="xs:string" name="name"/>
       <xs:element name="year" maxOccurs="unbounded" minOccurs="0">
        <xs:complexType>
         <xs:sequence>
          <xs:element type="xs:short" name="value"/>
          <xs:element name="month" maxOccurs="unbounded" minOccurs="0">
           <xs:complexType>
            <xs:sequence>
             <xs:element type="xs:short" name="value"/>
             <xs:element name="row" maxOccurs="unbounded" minOccurs="0">
              <xs:complexType>
               <xs:sequence>
                <xs:element type="xs:int" name="id"/>
                <xs:element name="attributes">
                 <xs:complexType>
                  <xs:sequence>
                   <xs:element name="attribute" maxOccurs="unbounded" minOccurs="0">
                    <xs:complexType>
                     <xs:sequence>
                      <xs:element type="xs:string" name="name"/>
                      <xs:element type="xs:string" name="value"/>
                     </xs:sequence>
                    </xs:complexType>
                   </xs:element>
                  </xs:sequence>
                 </xs:complexType>
                </xs:element>
                <xs:element type="xs:string" name="comments" maxOccurs="1" minOccurs="0"/>
                <xs:element type="xs:string" name="additional_comments" maxOccurs="1" minOccurs="0"/>
                <xs:element type="xs:float" name="amount"/>
                <xs:element type="xs:string" name="unit"/>
               </xs:sequence>
              </xs:complexType>
             </xs:element>
```

```
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

This is a XML example:

```
<?xml version='1.0' encoding='UTF-8'?>
<timephased>
  <object>82</object>
  <timephased_attribute>363</timephased_attribute>
  <rate_type>0</rate_type>
  <path>IT Portfolio</path>
  <name>PRJ 001</name>
  <version>
    <name>Forecast</name>
    <year>
      <value>2018</value>
      <month>
        <value>1</value>
        <row>
          <id>2</id>
          <attributes>
            <attribute>
              <name>ATTRID_346</name>
              <value>1.Europe</value>
            </attribute>
            <attribute>
              <name>ATTRID_362</name>
              <value>1.Labor Cost</value>
            </attribute>
          </attributes>
          <comments>This is a comment</comments>
          <additional_comments>This is an additional comment</additional_comments>
          <amount>1000</amount>
          <unit>Euro</unit>
        </row>
      </month>
      <month>
        <value>2</value>
        <row>
          <id>3</id>
          <attributes>
            <attribute>
```

```
                <name>ATTRID_346</name>
                <value>1.Europe</value>
              </attribute>
              <attribute>
                <name>ATTRID_362</name>
                <value>1.Labor Cost</value>
              </attribute>
            </attributes>
            <comments>This is a comment</comments>
            <additional_comments>This is an additional comment</additional_comments>
            <amount>1000</amount>
            <unit>Euro</unit>
          </row>
        </month>
      </year>
      <year>
        <value>2019</value>
        <month>
          <value>1</value>
          <row>
            <id>4</id>
            <attributes>
              <attribute>
                <name>ATTRID_346</name>
                <value>1.Europe</value>
              </attribute>
              <attribute>
                <name>ATTRID_362</name>
                <value>1.Labor Cost</value>
              </attribute>
            </attributes>
            <amount>1000</amount>
            <unit>Euro</unit>
          </row>
        </month>
        <month>
          <value>2</value>
          <row>
            <id>5</id>
            <attributes>
              <attribute>
                <name>ATTRID_346</name>
                <value>1.Europe</value>
              </attribute>
              <attribute>
                <name>ATTRID_362</name>
                <value>1.Labor Cost</value>
              </attribute>
            </attributes>
            <amount>1000</amount>
            <unit>Euro</unit>
          </row>
        </month>
      </year>
    </version>
  </timephased>
```

Here is the description for some of these fields:

- object: Object unique identifier provided by Triskell.
- timephased_attribute: Attribute unique identifier provided by Triskell.
- rate_type: Rate type unique identifier provided by Triskell.
- path: Complete path for a custom component. Separator "/".
- version name: case sensitive using the name of the version in Triskell.
- row: unique identifier by request.
- unit: case sensitive with same options than on Triskell.
- Attributes: name and value can be taken from Triskell custom attributes definition or from tenant_vw_attributes_list view on reporting module.
- Comments: this option is available when the timephased attribute is “Multi line with comments”. It’s optional but it’s part of the key for a timephased row.
- Additional comments: this option is available when the timephased attribute is “Additional comment”. It’s optional and it’s not part of the key and it should be included for the same timephased row when needed.

This is a response XML example:

```
<?xml version='1.0' encoding='UTF-8'?>
<timephased>
  <data>
    <result>OK</result>
    <row>1</row>
  </data>
  <data>
    <result>OK</result>
    <row>2</row>
  </data>
  <data>
    <result>KO </result>
    <row>3</row >
    <error>Invalid Name and Path</error>
  </data>
</timephased>
```

This web service will override all version data for each object

Information should be shorted by version, year and month

No Java Client is provided to use this service

Excel example is provided on the documentation

5 Service Proxies

Service Proxies are a simple way of executing remote procedure calls on Triskell CRUD and 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":""
}
```

5.1 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.

5.1.1 Stateful Operation Service Proxy

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

5.1.2 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

5.1.3 Stateless Operation Service Proxy

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

5.1.4 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

5.1.5 Java Client

The Triskell ServiceOperation client for JAVA will be provided into an Eclipse project containing all the needed classes and dependencies.

Package	eu.triskell.ws.rest.client.businessDelegate.operationService
Class	OperationServiceClient

Constructor Summary
OperationServiceClient(Conversation conv)

Method Summary	
DataResult	execute(java.lang.String serviceName, java.lang.String operationName)
DataResult	execute(java.lang.String serviceName, java.lang.String operationName, DataRequest request)

This is an example on how to execute a ServiceOperation call from JAVA:

```
Conversation conversation = null;
DataResult res  = null;

try {
        conversation = new Conversation("https://ondemand.triskellsoftware.com");
        conversation.logOn("admin@tenant.com", "aeiou");
} catch (TkRestClientException e) {
        e.printStackTrace();
        return;
}

try{
        OperationServiceClient osc = new OperationServiceClient(conversation);

        Map<String, String> params = new HashMap<String, String>();
```

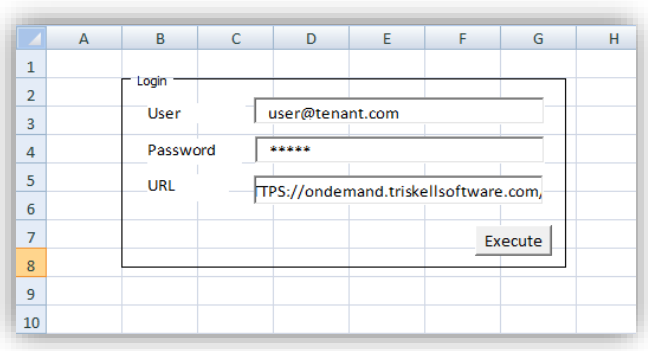
```
            params.put("PARAM_1", "X");


            DataRequest dr = new DataRequest(params, 0);
            res = osc.execute("ExampleService", "ExampleOperation", dr);

} catch (TkRestClientException e) {
            e.printStackTrace();
}catch (Throwable e) {
            e.printStackTrace();
} finally {
            try {
                        conversation.logOff();
            } catch (TkRestClientException e) {
                        e.printStackTrace();
                        return;
            }
}
```

5.1.6 VBA Client

The Triskell ServiceOperation client for VBA will be provided into an Excel document containing all the needed classes and modules.



This is an example on how to execute a ServiceOperation call from VBA:

```
Dim osclient As New TkOperationServiceClient
Dim dataRequest As New TkDataRequest
Dim result As Dictionary

dataRequest.AddParameter "PARAM_1", "A"
dataRequest.AddParameter "PARAM_2", "B"

Set result = osclient.ExecuteWithData(pService:="ServiceName", _
                                      pOperation:="OperationName", _
                                      pDataRequest:=dataRequest)
```

6 Operational Services

6.1 Data Export (only data)

To export data from Triskell you need to use the reporting web service.

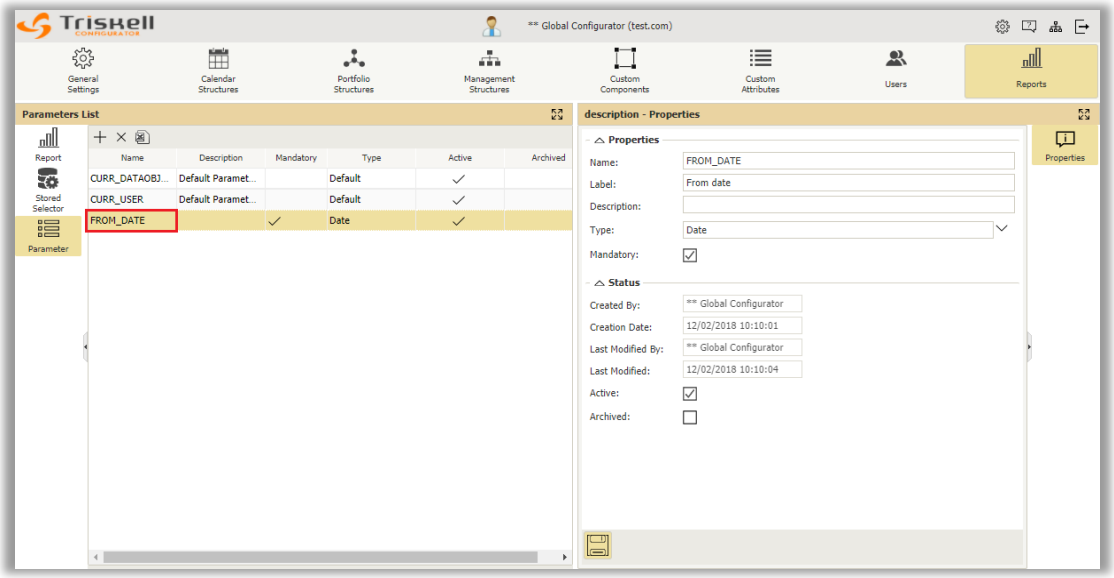
It's a proxied service (Please read documentation above). It requires previous standard authentication, using the login web service.

We are going to describe an example showing how to use this reporting service to extract data from Triskell easily.

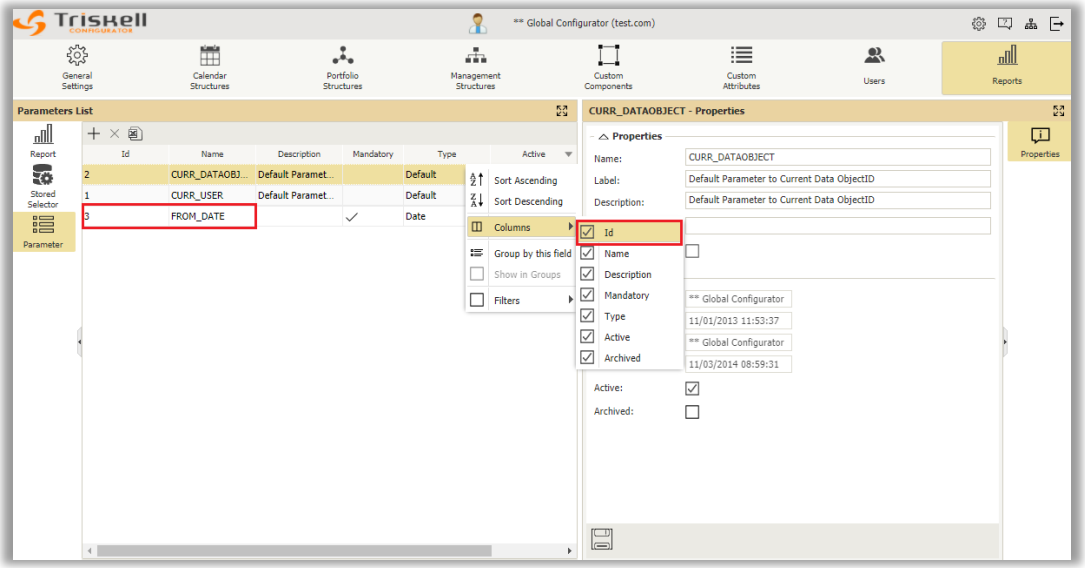
First, you need to build a stored selector in Triskell. It's not needed to add it to a report.

In this example, we are going to extract projects data that has been modified after a given date.

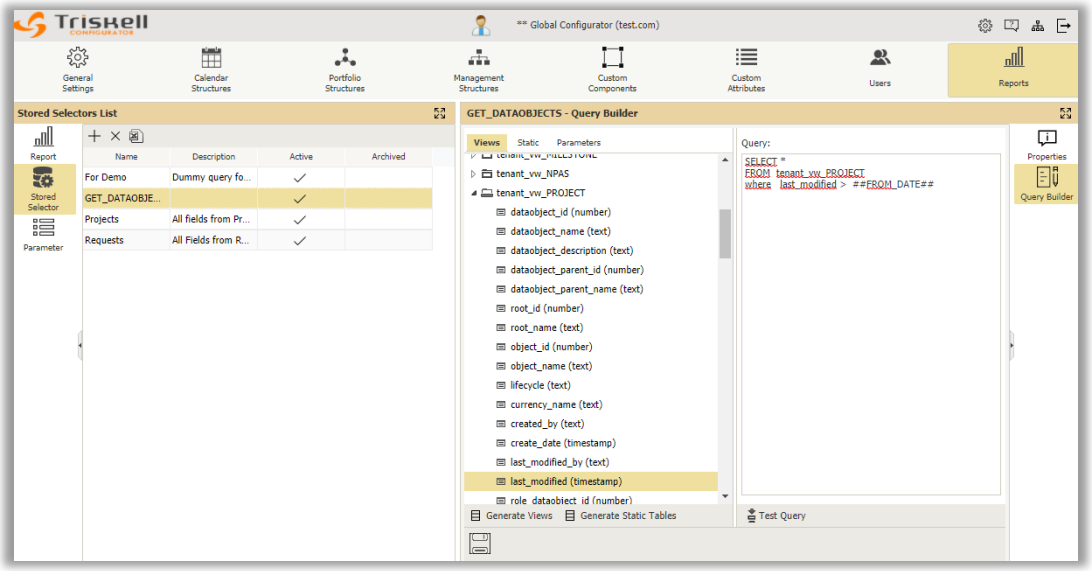
<u>First, we need to create the parameter on Triskell reporting module:</u>



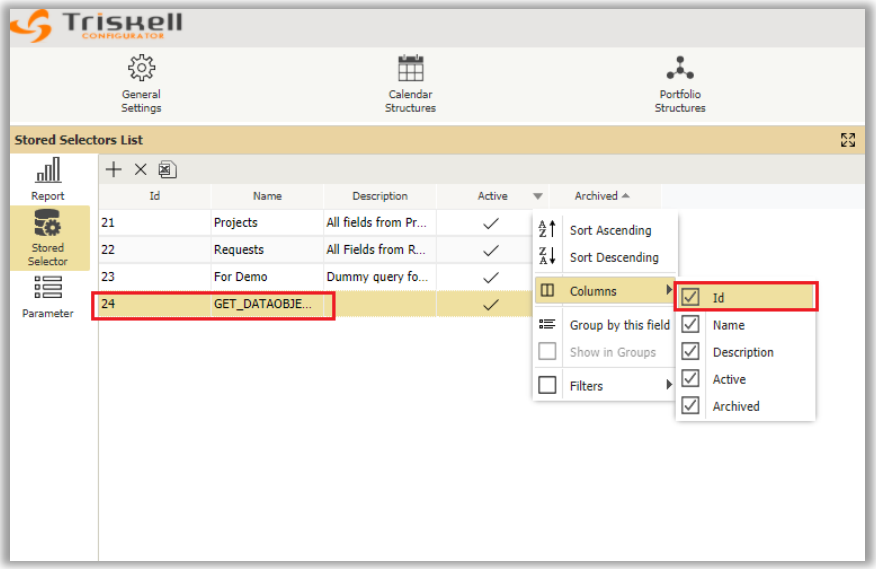
We need this parameter id to add to our web service request so we need to display Id column:



Then we create a stored selector using these parameters:



We need to get the stored report id to run the query from a web service, so display the id column:



This is the query we use to build the stored selector:

```
SELECT *
FROM  tenant_vw_PROJECT
where   last_modified >  ##FROM_DATE##
```

URL	URL: https:// SERVER /triskell/service/rest/proxy/operation/execute/ReportService/GetData
HTTP Method	POST
Content-Type Header	application/json

URL Parameters	JSON format

JSON request example:

```
{
        'id' : 0  ,
        'params' : {
                "STORED_SELECTOR_ID":"24",
                "valuesByParams":"3#2018.02.01"
        } ,
        'objects' : null
}
```

Field description:

- id: 0. It's required but not used for this service.
- objects: null. It's required but not used for this service.
- params:
    - STORED_SELECTOR_ID: Required. It's your stored selector Id taken from Triskell.
    - valuesByParams: Contains all report parameters. For each parameter the format is: {paramId}#{value}. You can concatenate several report params using "##" separator.

Example of JSON result:


```
{
  "authash": "03663e556a63170e8fe3b40667f6e09f",
  "executime": 118761151,
  "success": true,
  "messages": [],
  "message": null,
  "ret_i18n_code": "",
  "ret_i18n_params": [],
  "trace": null,
  "resultType": 1,
  "data": {
    "isTruncatedResult": false,
    "res": [
      {
        "$_$OwningDepartment": "AIIBE",
        "$FCBudgetd": 127,
        "$_$Successor": "",
        "$_$Administrator": "* IT PMO",
        "create_date": "11/13/2012",
        "lifecycle": "5.Deployment",
        "$ResProgress": 78.8,
        "$ImpactBusOps": "High",
        "$Platform": "Client-Server",
        "$ImplementationRisk": "Medium",
        "created_by": "** Global Configurator",
        "$ITScorecard": "1.Corporate Contribution;3.Operational Excellence",
        "dataobject_description": "Converse Platinum Xenon",
        "$PoolActuals": 1502,
        "$_$ProjectMembers": "07.Sean Pearson (IT User);Milo T. Riley;Tomas Osterhagen;Bernadette Hébert;Nicholas Richardson;Leo Watkins",
        "role_dataobject_id": 1064,
        "$Budgetk": 5620,
        "$FCBudgetk": -530,
        "$_$Application": "A-RAVEN",
        "$_$ProjectSponsor": "",
        "$Fin": "4.Some Issues",
        "dataobject_name": "PRJ 001",
        "$PoolAllocated": 5657.25,
        "$Baselinestart": "",
        "$Baselinefinish": "",
        "dataobject_id": 1064,
        "$_$Client": "Finance",
        "root_name": "IT Portfolio",
        "$Bus": "2.In Progress",
        "$_$Initiatives": "BI 2015-001; BI 2015-002; BI 2015-004",
        "root_id": 1025,
        "$InitiativeCost": "High",
        "$Finishdate": "12/31/2018",
        "$Budgetd": 885,
        "$Agile": "No",
        "$PoolTaskAssignment": 6496.16,
        "$Forecastd": 1012,
        "last_modified": "02/13/2018",
        "last_modified_by": "** Global Configurator",
        "$Region": "1.Europe",
        "rel_dataobject_id": 1064,
        "$_$ProjectManager": "** Global Configurator;02.Uwe Seiler (PM)",
        "$Forecastk": 5090,
        "$ExpBenefits": 500,
        "dataobject_parent_id": 1025,
        "$FinProgress": 76.87,
        "$Progress": "050",
        "$Probability": 25,
        "$Actualsd": 697.38,
        "$Score": 5.8,
        "$Weightedscore": 1.45,
        "object_id": 82,
        "$Tech": "1. On Top",
        "$Actualsk": 4320,
        "$ActivityType": "1. Corporate Projects",
        "$StrategicImpact": "High",
        "attr_dataobject_id": 1064,
```

```
        "dataobject_parent_name": "IT Portfolio",
        "currency_name": "Euro",
        "object_name": "Project",
        "$Status": "Excellent",
        "$complete": 1.246737,
        "$Risk": "3.Stable",
        "$ImpactBusStrategy": "High",
        "$StartDate": "10/02/2017",
        "$_$FinancialController": "10.Samsa Leskinen (IT Fin.)"
    },
    {
        "$_$OwningDepartment": "AIIBE",
        "$FCBudgetd": -162.5,
        "$_$Successor": "PRJ 011",
        "$_$Administrator": "",
        "create_date": "11/13/2012",
        "lifecycle": "4.Testing",
        "$ResProgress": 130.77,
        "$ImpactBusOps": "High",
        "$Platform": "Client-Server",
        "$ImplementationRisk": "Low",
        "created_by": "** Global Configurator",
        "$ITScorecard": "3.Operational Excellence",
        "dataobject_description": "Ralph Aluminium Stinger",
        "$PoolActuals": 0,
        "$_$ProjectMembers": "07.Sean Pearson (IT User);Milo T. Riley;Tomás Barros Silva;Bernadette Hébert",
        "role_dataobject_id": 1065,
        "$Budgetk": 2000,
        "$FCBudgetk": -500,
        "$_$Application": "A-RAVEN",
        "$_$ProjectSponsor": "01.Bruce H. Trahan (IT Exec)",
        "$Fin": "5.Critical",
        "dataobject_name": "PRJ 002",
        "$PoolAllocated": 1460,
        "$Baselinestart": "07/01/2017",
        "$Baselinefinish": "12/31/2018",
        "dataobject_id": 1065,
        "$_$Client": "Finance",
        "root_name": "IT Portfolio",
        "$Bus": "4.Some Issues",
        "$_$Initiatives": "BI 2015-026",
        "root_id": 1025,
        "$InitiativeCost": "Moderate",
        "$Finishdate": "12/31/2018",
        "$Budgetd": 162.5,
        "$Agile": "No",
        "$PoolTaskAssignment": "",
        "$Forecastd": 0,
        "last_modified": "02/13/2018",
        "last_modified_by": "** Global Configurator",
        "$Region": "1.Europe",
        "rel_dataobject_id": 1065,
        "$_$ProjectManager": "02.Uwe Seiler (PM)",
        "$Forecastk": 1500,
        "$ExpBenefits": 600,
        "dataobject_parent_id": 1025,
        "$FinProgress": 0,
        "$Progress": "080",
        "$Probability": "",
        "$Actualsd": 212.5,
        "$Score": 7.8,
        "$Weightedscore": 0,
        "object_id": 82,
        "$Tech": "1. On Top",
        "$Actualsk": 0,
        "$ActivityType": "2.Bus. Improvements",
        "$StrategicImpact": "Medium",
        "attr_dataobject_id": 1065,
        "dataobject_parent_name": "IT Portfolio",
        "currency_name": "Euro",
        "object_name": "Project",
        "$Status": "Excellent",
        "$complete": 0,
        "$Risk": "4.Some Issues",
```

```
            "$ImpactBusStrategy": "Low",
            "$StartDate": "07/01/2017",
            "$_$FinancialController": ""
        },
        {
            "$_$OwningDepartment": "DWDI",
            "$FCBudgetd": -62.5,
            "$_$Successor": "",
            "$_$Administrator": "",
            "create_date": "11/13/2012",
            "lifecycle": "3.Development",
            "$ResProgress": 80,
            "$ImpactBusOps": "Low",
            "$Platform": "DB Services",
            "$ImplementationRisk": "Low",
            "created_by": "** Global Configurator",
            "$ITScorecard": "1.Corporate Contribution",
            "dataobject_description": "Sphere Heliotrope Plato",
            "$PoolActuals": "",
            "$_$ProjectMembers": "",
            "role_dataobject_id": 1080,
            "$Budgetk": 700,
            "$FCBudgetk": 100,
            "$_$Application": "KRONOS",
            "$_$ProjectSponsor": "",
            "$Fin": "3.Stable",
            "dataobject_name": "PRJ 017",
            "$PoolAllocated": "",
            "$Baselinestart": "07/01/2017",
            "$Baselinefinish": "02/28/2019",
            "dataobject_id": 1080,
            "$_$Client": "Strategy",
            "root_name": "IT Portfolio",
            "$Bus": "1.On Top",
            "$_$Initiatives": "BI 2015-028; BI 2015-030",
            "root_id": 1025,
            "$InitiativeCost": "Moderate",
            "$Finishdate": "02/28/2019",
            "$Budgetd": 62.5,
            "$Agile": "No",
            "$PoolTaskAssignment": "",
            "$Forecastd": 0,
            "last_modified": "02/09/2018",
            "last_modified_by": "** Global Configurator",
            "$Region": "3.Japan",
            "rel_dataobject_id": 1080,
            "$_$ProjectManager": "",
            "$Forecastk": 800,
            "$ExpBenefits": 700,
            "dataobject_parent_id": 1025,
            "$FinProgress": 50,
            "$Progress": "010",
            "$Probability": "",
            "$Actualsd": 50,
            "$Score": 7.4,
            "$Weightedscore": 0,
            "object_id": 82,
            "$Tech": "2.In Progress",
            "$Actualsk": 350,
            "$ActivityType": "1. Corporate Projects",
            "$StrategicImpact": "Medium",
            "attr_dataobject_id": 1080,
            "dataobject_parent_name": "IT Portfolio",
            "currency_name": "Euro",
            "object_name": "Project",
            "$Status": "Correct",
            "$complete": 15,
            "$Risk": "2.In Progress",
            "$ImpactBusStrategy": "High",
            "$StartDate": "07/01/2017",
            "$_$FinancialController": ""
        }
    ]
},
```

```
  "description": "GET_DATAOBJECTS#$#3,0.075",
  "id": 0,
  "ret_count": 0,
  "i18NParams": [],
  "i18NMessageId": ""
}
```


Reporting Web Service has a default rows limitation (1000 rows) to avoid performance issues.
It is configurable parameter but you can also rethink your approach to do more request and getting smaller set of data.

6.2 Data Export (with data definition)

To export data from Triskell you need to use the reporting web service.

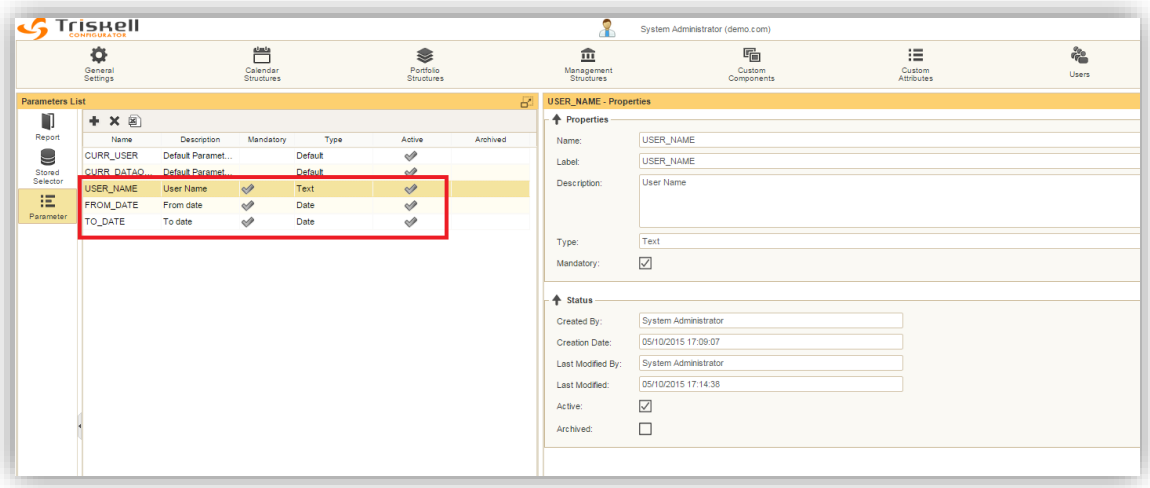
It's a proxied service (Please read documentation above). It requires previous standard authentication, using the login web service.

We are going to describe an example showing how to use this reporting service to extract data from Triskell easily.

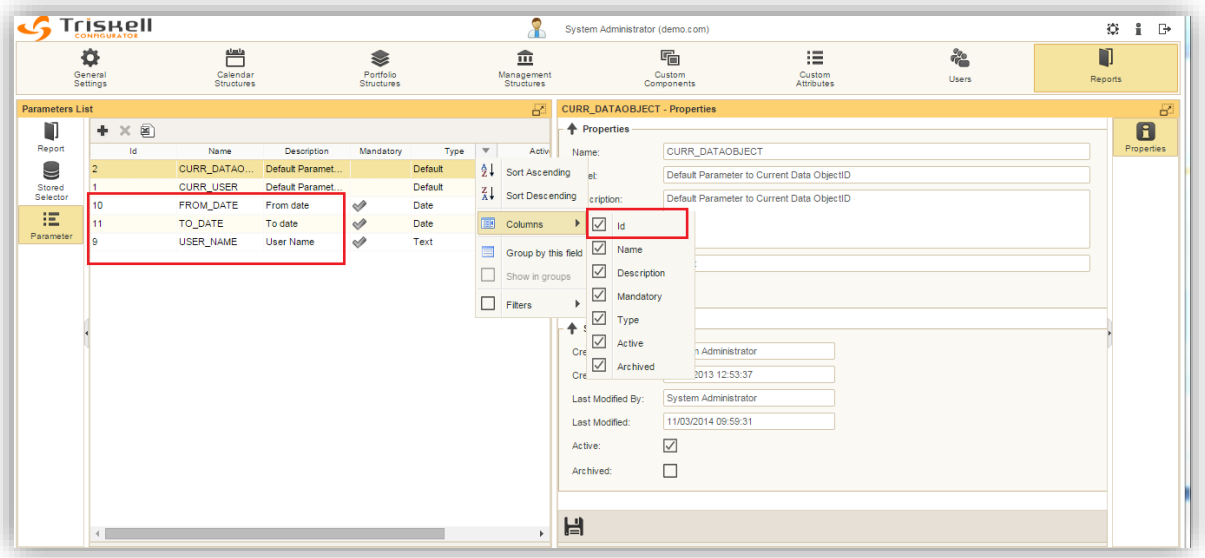
First, you need to build a stored selector in Triskell and add it to a report. You can only assign one stored selector to your report.

In this example, we are going to extract timesheet data for a given user between 2 dates.

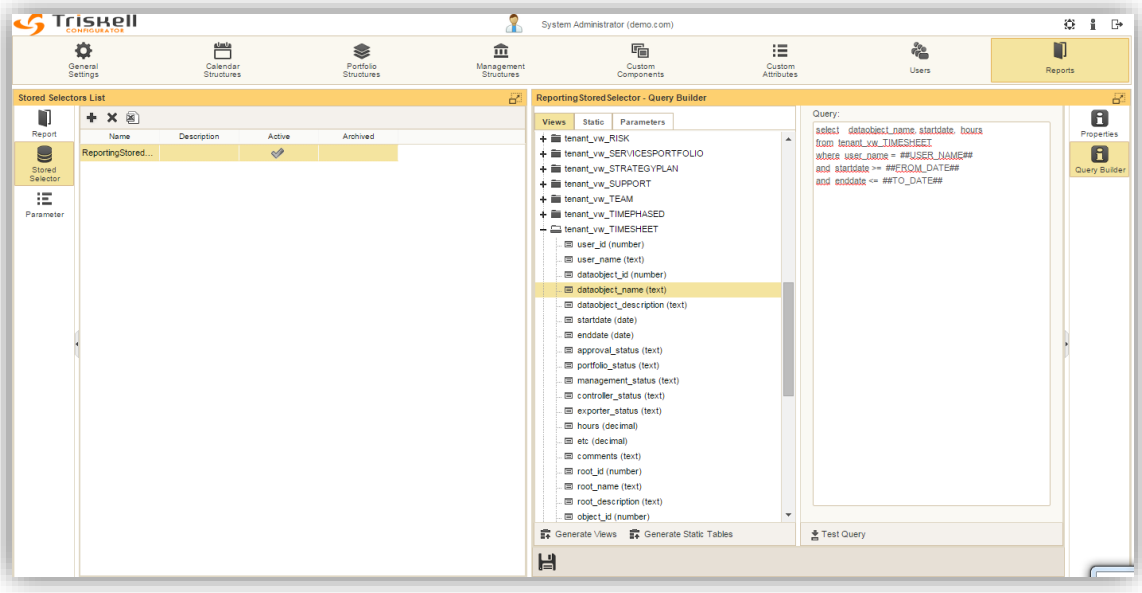
<u>First we need to create the parameters on Triskell reporting module:</u>



We need these parameter ids to add to our web service request so we need to display Id column:



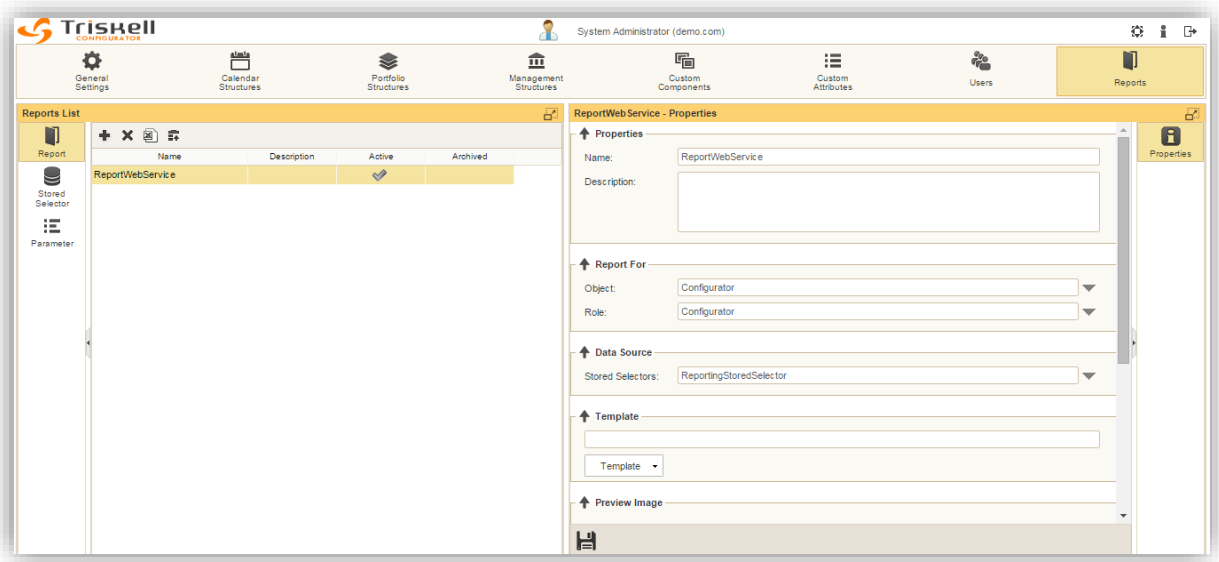
Then we create a stored selector using these parameters:



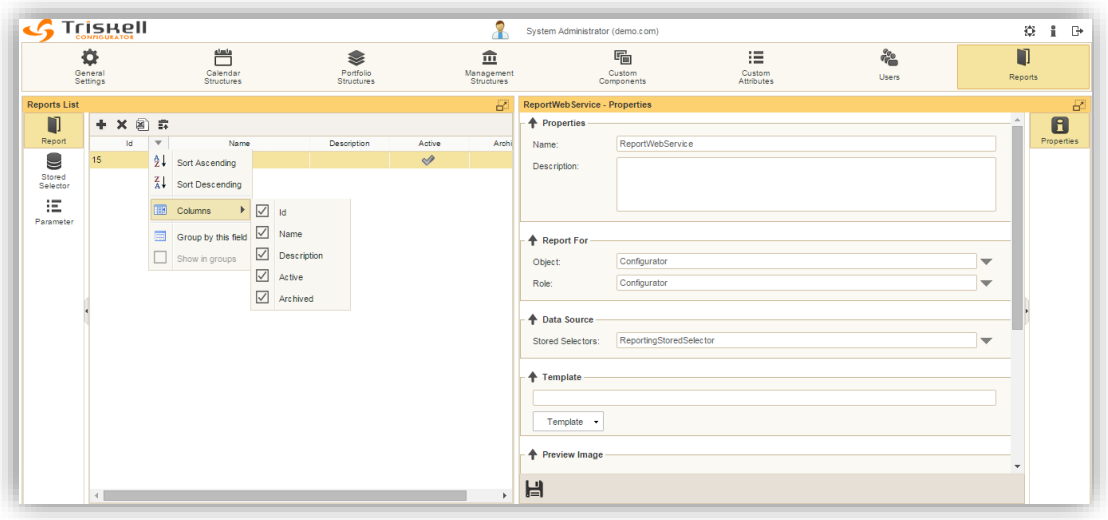
This is the query we use to build the stored selector:

```
select   dataobject_name, startdate,  hours
from  tenant_vw_TIMESHEET
where  user_name =  ##USER_NAME##
and  startdate >=  ##FROM_DATE##
and  enddate <=  ##TO_DATE##
```

Then we need to create the report:



We need to get the report id to run the report from a web service, so display the id column:



URL	URL: https:// SERVER /triskell/service/rest/proxy/operation/execute/ReportService/GetReportDataToPanel
HTTP Method	POST
Content-Type Header	application/json
URL Parameters	JSON format

JSON request example:

```
{
        'id' : 0 ,
        'params' : {
                "REPORTID":"15",
                "valuesByParams":"9#System Administrator##10#2015.10.5##11#2015.10.11"
        } ,
        'objects' : null
}
```

Field description:

- id: 0. It's required but not used for this service.
- objects: null. It's required but not used for this service.
- params:
    - REPORTID: Required. It's your report Id taken from Triskell.
    - valuesByParams: Contains all report parameters. For each parameter the format is: {paramId}#{value}. You can concatenate several report params using "##" separator.

Example of JSON result:


```
{
  "authash": "dd06d7c94cd9a73f62b45c5b54247bce",
  "executime": 181231719,
  "success": true,
  "messages": [],
  "message": null,
  "resultType": 1,
  "data": {
    "ReportingStoredSelector": {
      "isTruncatedResult": false,
      "res": [
        {
          "startdate": "10/05/2015",
          "hours": 2,
          "dataobject_name": "Integración de los componentes"
        },
        {
          "startdate": "10/06/2015",
          "hours": 2,
          "dataobject_name": "Create new Risk"
        }
      ],
      "columns": [
        {
          "text": "dataobject_name",
          "dataIndex": "dataobject_name",
          "type": "string",
          "filter": {
            "type": "string"
          }
        },
        {
          "text": "startdate",
          "xtype": "datecolumn",
          "dataIndex": "startdate",
          "format": "Ext.Date.defaultFormat",
          "filter": {
            "type": "date"
          }
        },
        {
          "text": "hours",
          "dataIndex": "hours",
          "type": "float",
          "filter": {
            "type": "numeric"
          }
        }
      ],
      "fields": [
        {
          "name": "dataobject_name",
          "type": "string"
        },
        {
          "name": "startdate",
          "type": "date"
        },
        {
          "name": "hours",
          "type": "float"
        }
      ]
    }
  },
  "id": 0,
  "i18NMessageId": "",
  "i18NParams": []
}
```

Reporting Web Service has a default rows limitation (1000 rows) to avoid performance issues.
It is configurable parameter but you can also rethink your approach to do more request and getting smaller set of data.

6.3 Advanced Lifecycle

It's a proxied service (Please read documentation above). It requires previous standard authentication, using the login web service.

We are going to describe an example showing how to use this service to transition a dataobject from the current state to a new one.

First, you need to build a stored selector and a report in Triskell to obtain the dataobject and stage id's, as described on the previous section 6.1 Data Export.

URL	https://SERVER/triskell/service/rest/proxy/operation/execute/**ObjectPropertiesPanelAS/AdvancedLifecycleChange**
HTTP Method	POST
Content-Type Header	application/json
URL Parameters	JSON format

JSON request example:

```
{
  "objects":"",
  "params": {
    "dataObjectId":"1064",
    "newLifecycle":"62",
    "commentToTransition":"Text on commentToTransition",
    "attachmentsToTransition":"Text on attachmentsToTransition"
  },
  "id":0
}
```

Field description:

- id: 0. It's required but not used for this service.
- objects: null. It's required but not used for this service.
- params:
  - dataObjectId : Required. It's your dataobject Id taken from Triskell.
  - newLifecycle : Required. It's the new state on your dataobject Id taken from Triskell.
  - commentToTransition : Comment about the transition.
  - attachmentsToTransition : Text to attach to the transition.

Example of successful JSON result:

```
{
 "authash": "03663e556a63170e8fe3b40667f6e09f",
 "executime": 233380846,
 "success": true,
 "messages": [],
 "message": null,
 "resultType": 1,
```

```
  "i18NMessageId": "",
  "i18NParams": []
}
```


Example of error JSON result:


```
{
  "authash": "03663e556a63170e8fe3b40667f6e09f",
  "executime": 0,
  "success": false,
  "messages": [],
  "message": "Transition to stage 2.Analysis not allowed",
  "resultType": 3,
  "i18NMessageId": "",
  "i18NParams": []
}
```