SV Web Services Toolkit REST API

Developer's manual

Last update :	2023-08-22
Revision :	1.0
Content :	This document describes the REST Interface of SV Web Services Toolkit.
Confidentiality :	Restricted

PcVue® Solutions

Content

Scope of this document

The scope of this document is to describe and explain functional aspects of the SV *Web Services Toolkit RESTful API*. It is targeted at developers willing to implement a Web Services client taking advantage of the SV *Web Services Toolkit* public REST interfaces.

Web Services concepts

1. SV Web Services Toolkit

SV is intended to be a **web services server**.

Web Services Toolkit allows third-party applications to exchange data with the supervisor. It is a server API (Application Programming Interface) that provides following accesses:

- Authentication/session management
- Realtime data access
- Real time alarm access
- Historical data access: logged events and trends
- Graphical data: mimics and symbols

Web Services Toolkit is available for several technologies:

- SOAP/XML since SV version 8.00
- REST since SV version 12.0

Present document exclusively focuses on the RESTful interface, which contains 5 public web services named:

- OAuth
- RealtimeData
- RealtimeAlarm
- HistoricalData
- GraphicalData (not part in this document)

For the description of SV SOAP/XML web services, see the corresponding documentation.

Architecture

The set of web services contained in the *Web Services Toolkit* REST are only accessible over HTTPS (http secured).
SV relies on the Microsoft Internet Information Server (IIS) to manage the http layer. The IIS compatible versions with *Web Services Toolkit* follow the Operating Systems compatible with SV.

In SV architecture, one of the SCADA stations can have the role of *Web Backend*, which communicates (locally or remotely) with the Web application server (IIS). To set up web server, it is necessary to use the Web Deployment Console (WDC) provided with SV.

See SV documentation or online help for more information

- about Windows OS compatibility;
- about the *Web Services Toolkit* pre-requisite and IIS installation;
- about the web server deployment and WDC.

Licensing

The *Web Services Toolkit* is licensed. To be able to have the Web application server answering to calls, a licence is needed with at least one “Web Services Toolkit client” allowed. One session consumes one connection.

2. Web Services REST

A Web Service is an application or a data source reachable through a standard Web protocol, which allows communication and data exchange between heterogeneous applications over a network. A Web Service contains functionalities which are exposed from a Web server by using a communication based on requests-answers principle.

Web Services **REST** (REpresentational State Transfer) are lightweight, maintainable, and scalable services based on HTTP and URI standards. They fully expose their functionalities as a set of resources identifiable by URI and available through HTTP protocol syntax and semantics.

The key elements of a RESTful implementation are:

- the **Resources** – data to be reached.
  The address of a resource must be directly passed in the URL.
  Example: in *RealtimeData* web service, the value of a variable "Branch01.Register01" is reachable with at
  https://[ServerAddress]/RealtimeData/v2/Values/Branch01/Register01

- the **Request Verbs** – access methods which describe what to do with the resource.
  The standard methods are GET (to read), POST (to add), PUT (to update), PATCH (to modify) and DELETE (to remove)

- the **Request Headers** – additional instructions sent with the request.
  These might define the type of response required or the authorization details.

- the **Request Body** – data to be sent with the request.
  Mainly used for POST messages, when data have to be added or modified.

- the **Response Body** – this is the main body of the response.
  The format of the response (XML, JSON, text…) can be defined in the request, according to the request type and the server capability.
  The default response format for the *Web Services Toolkit* is JSON. But XML format can also be used.

- the **Response Status codes** – general codes which are returned with the response.
  For example, the code 200 is normally returned if there is no error when returning a response to the client.

Naming rules

Naming rules used by the *Web Services Toolkit* REST follow best practices for RESTful services design. In order to distinguish them from the existing SOAP/XML interface (v1.0), all services are prefixed by "**/v2/**".

Example: https://[ServerAddress]/RealTimeData/v2/Variables/System

Resources

The *Web Services Toolkit* REST provides the following RESTful resources:

- Branches
- Variables
- Subscriptions
- Alarms
- Trends
- Logs
- Mimics
- Symbols

The access methods used are:

- GET: to ask for resource (variable read, get mimic, configuration …)
- POST: to send an action (write request, open session, create subscription)
- PUT: to update an existing resource (update subscription)
- DELETE: to delete user generated resources (like subscriptions)

Cybersecurity and Authentication

The *Web Services Toolkit* uses SSL/TLS encryption. It exclusively allows HTTPS communication.
The authorization is based on OAuth 2.0 **Bearer tokens**. This token is passed to the server in the "Authorization" HTTP header field.

The token is delivered by a specific authentication web service, which follows the state-of-the-art OAuth 2.0 specification, called "OAuth".

3. Basic principles

Web services take advantage of widely adopted technologies of the Internet field.
In particular, Http relies on a client-server architecture used in a not-connected mode e.g., no one can assume what happens on the other side between 2 requests.
More than that, the web services specification does not define anything with regards to session nor transaction management. Web services specifications are not designed to describe persistency between client requests.

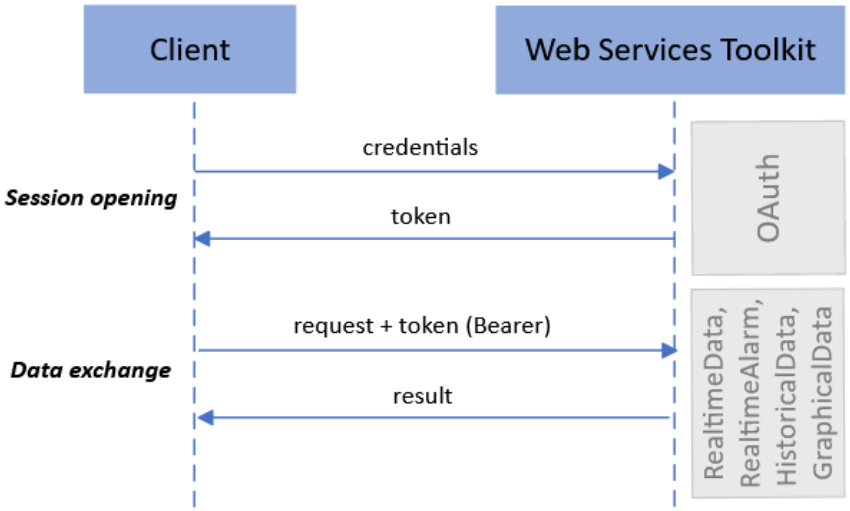


Figure 1 : communication principles

The *Web Services Toolkit* session layer gives SV the ability to:

- Manage data persistency between client requests.
- Associate resources allocated on the server side to the good client.
- Manage resources deallocation in case a client does not request anything for a long time (session time-out).

The session management layer permits a "connected mode" management and allows using the term "connected user": Once a session is open, and as soon as the client software calls the server from time to time, the session remains valid, and the server may hold persistent data for it.

3.1 Authentication

Based on OAuth 2.0, the authentication web service *OAuth* supports several authentication methods:

- Authorization code flow
- Password credential flow
- Refresh code flow

The result of authentication procedures is a message (by default formatted in JSON), containing:

- an **access token**: used for data exchanges
- a **refresh token**: used for session reactivation

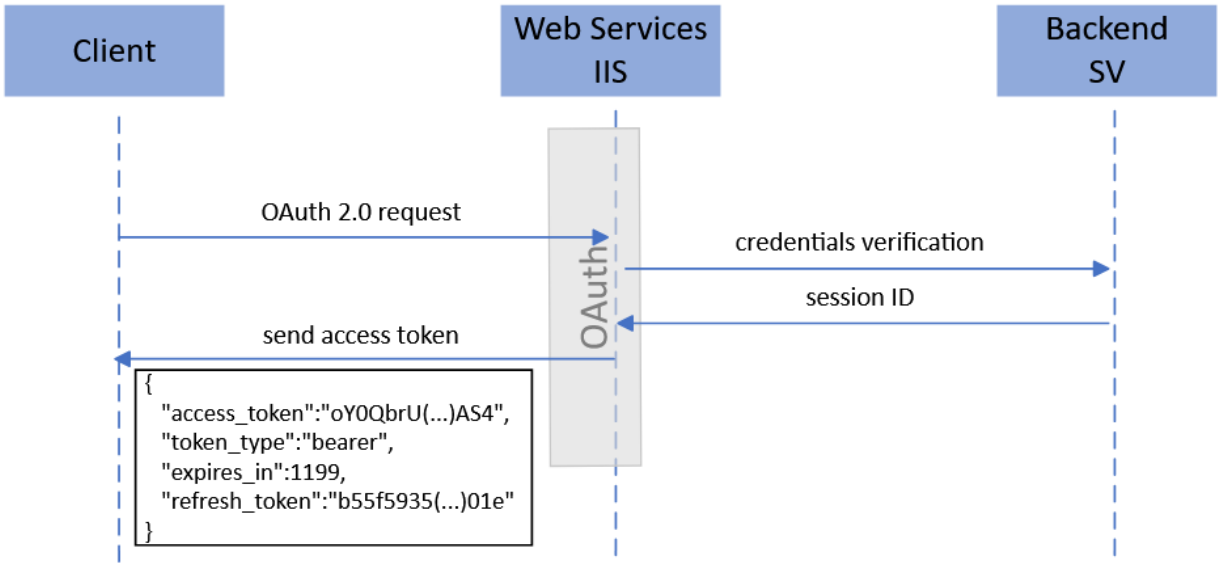


Figure 2 : authentication basic flow

3.2 Simple access to resource

Each web service allows to make a simple access to a resource. Those accesses are based on a one step request/response principle. The client only has to create an HTTP request with

- the right method (GET, POST)
- the good destination URI, according to the needed access
- a Header containing the authorization parameters (token)
- a Body, if necessary, containing the parameters to be sent

If the *token* is valid, the server answers returning a data.

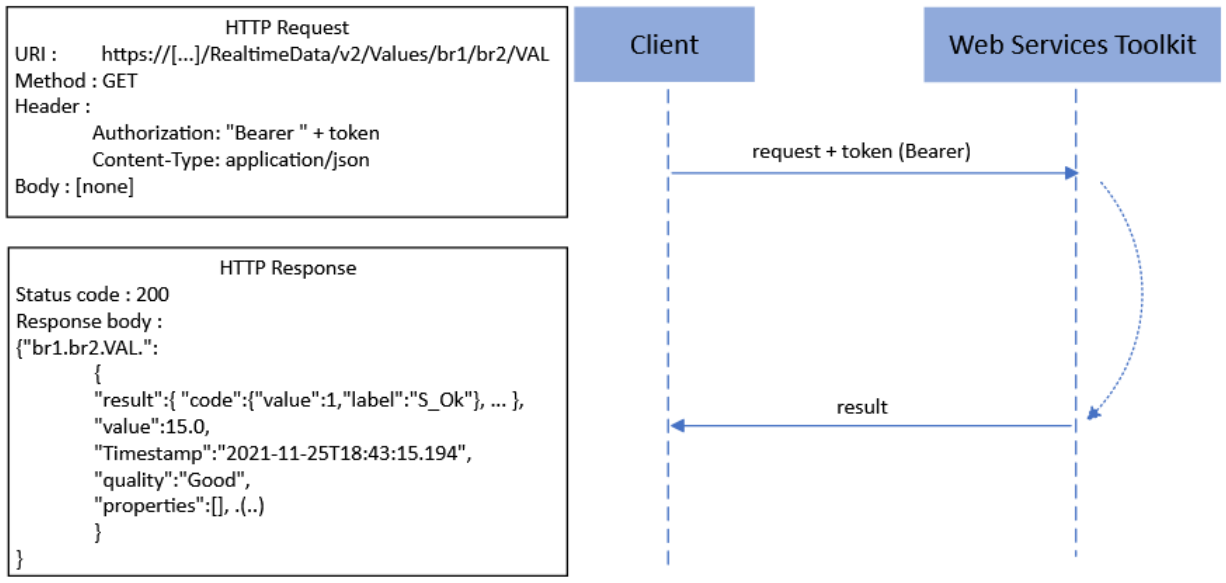


Figure 3 : Simple access principles - Read variable

For the *RealTimeData* service, the following simple accesses to real time variables are available:

- GET: Browse variables, Read variables, Read configuration
- POST: Write variables, Bulk read variables

For the *RealTimeAlarm* service, the following simple accesses to real time alarms are available:

- GET: Read list of alarms, Read filters
- POST: Send alarm associated action

For the *HistoricalData* service, only one access is available:

- GET: Read configuration, Get trend (method 2)

For the *GraphicalData* service, the following simple accesses to graphical objects are available:

- GET: Get a mimic, Get a symbol

3.3 Subscription to a resource

To read real time data from server, the *Web Services Toolkit* allows to use subscription to a resource. The subscription mechanism can be divided into 3 main steps.

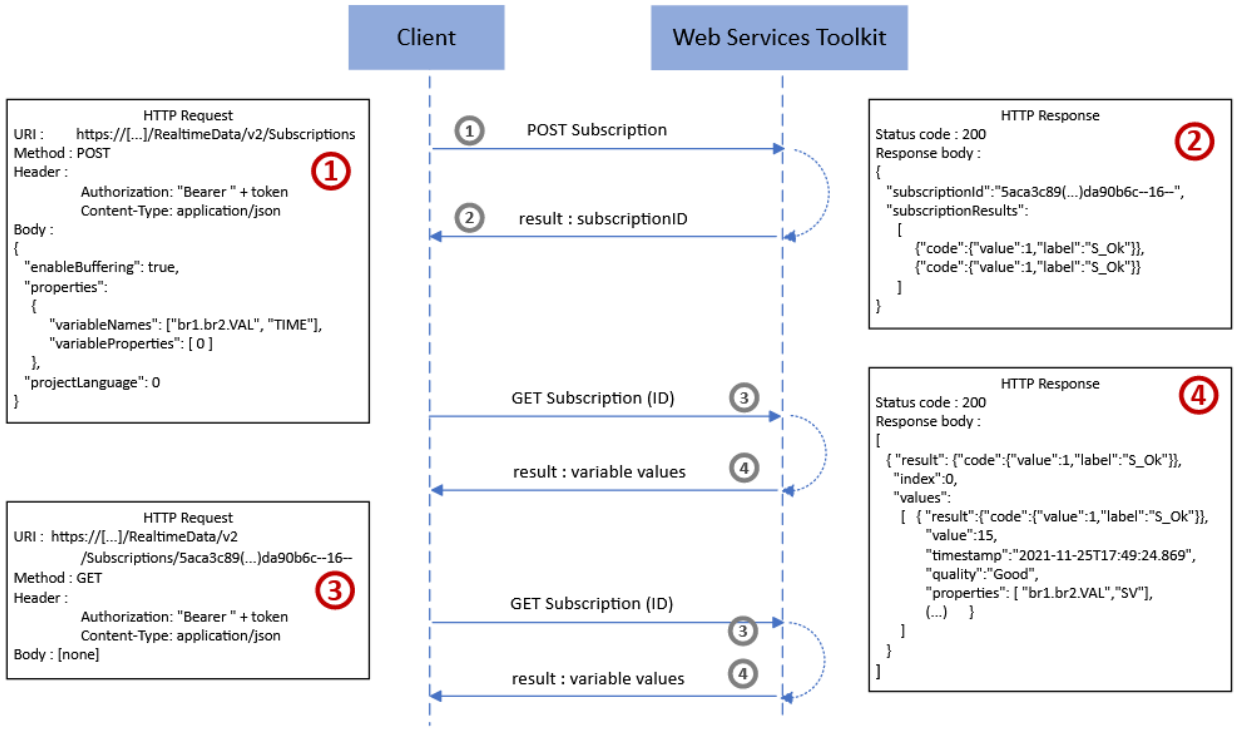


Figure 4 : Real time data subscription principles

Step 1:

As a preliminary operation, the web service client has to call the "subscription" method passing the server the parameters to subscribe (ex: a list of variable names). On receiving this request, the server allocates needed resources to be able to store data each time one of the requested resource changes.

Step 2:

When necessary for client-side processing, the client can call the "read subscribed data" method to get the set of data stored on server side since the last call (or since the subscription for the first call).

The call to "read subscribed data" method is usually cyclic or on user request on client side.

Step 3:

When the client does not need to retrieve values any more, the subscription can be cancelled by calling the "cancel subscription" method. Doing so allows the server to deallocate resources and stop watching for variable changes.

At any time during the life of such a subscription, the subscription parameters can be modified by calling the “modify subscription” method. In particular, it is possible to change the list of subscribed data.

Subscription options permit to retrieve only the last change of each data subscribed instead of all changes since last call to “read subscribed data” function.

The subscription mechanism is available for

- *RealTimeData* service: Read variables
- *RealTimeAlarm* service: Read Alarm lists

3.4 Two-step requests

To get historical data from server, it is necessary for SV to prepare the data which has to come from historical server.
The *Web Services Toolkit* allows to request SV archives for logged events and trends by using a two-step request. This kind of requests concerns only the *HistoricalData* service.

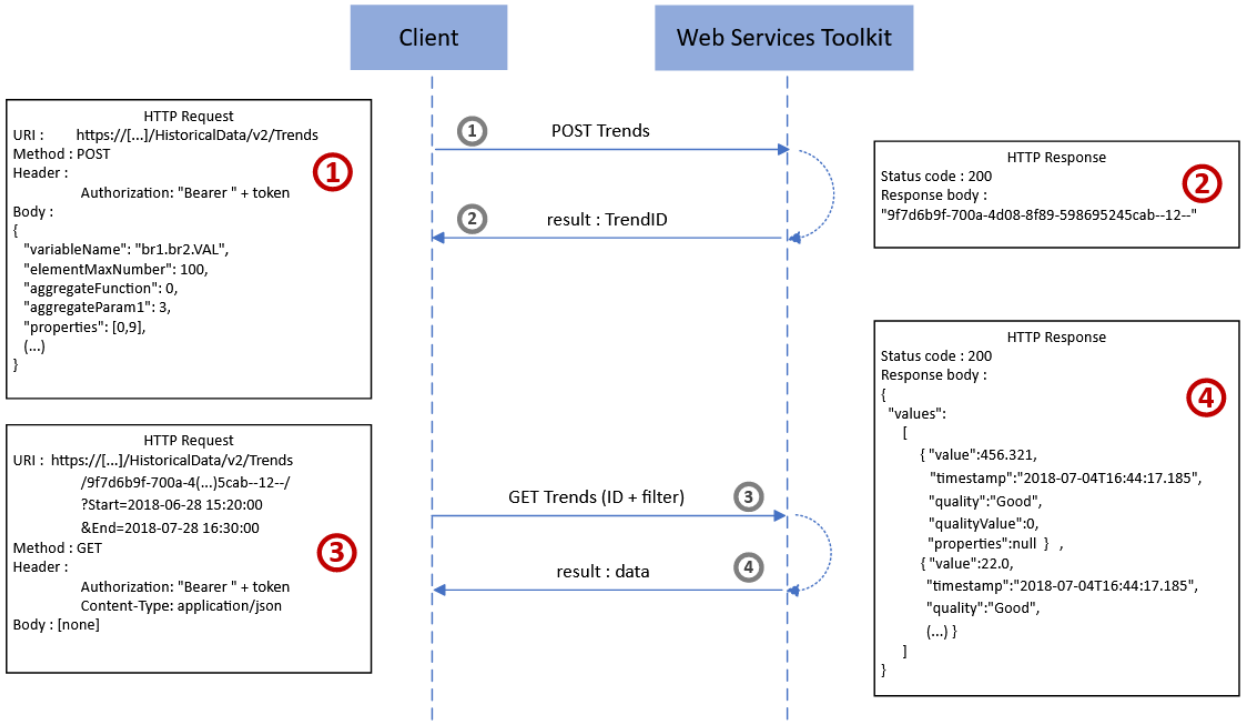


Figure 5 : Historical data request principles

The mechanism can be divided into 3 main phases:

Phase 1 (1st request: POST)

As a preliminary operation, the web service client has to call the "creation" method passing the server a set of parameters defining the request itself (log list name, trend name, additional filter, set of events and set of properties to retrieve in the SV archives). On receiving this request, the server allocates resources to be able to handle the request.

Phase 2 (2nd request: GET)

When necessary for client-side processing, the client can call the "get data" method to get the set of logged events / trend points for a given time period.

This request is usually called on user request on client side.

Phase 3

When the client does not need to retrieve logged events anymore, the log request can be closed by calling the "close" method. Doing so allows the server to deallocate resources.

At any time during the life of such a log request, the request parameters can be modified by calling the "modification" method. In particular, it is possible to change the filter.

The two-step requests are used in *HistoricalData* service for requesting:

- Trends: get values from archive
- Logs: get log list

3.5 Data

3.5.1 Timestamping

Data timestamps returned by the SV *Web Services Toolkit* methods or used as parameters are always of type *DateTime* and expressed in UTC.

3.5.2 Bilingual SV application

The *Web Services Toolkit* is able to return localized strings if it is available on server side.
The language to use for localized strings is defined by:

- The language defined in the profile of the user (for WebVue).
- The language defined at Session opening time.
- The language defined at method calling time.

See SVLanguage for more information.
See the SV documentation for more information on how to design bilingual applications.

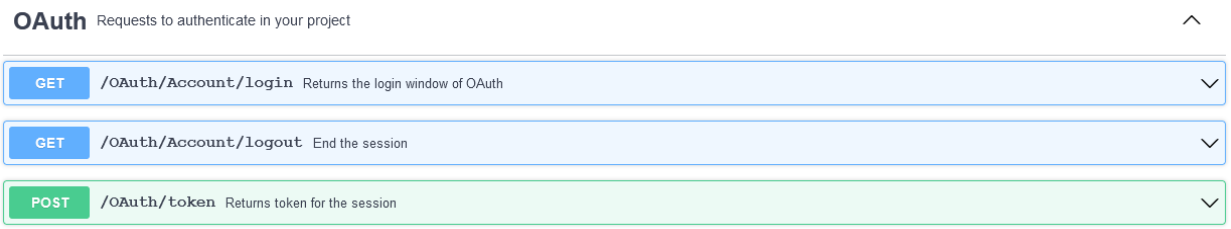
REST API description

1. Authentication service: OAuth

Accessing to SV data requires that a session is opened between the client and the server.

Through web service ***OAuth***, the following operations are possible:

- Request login window (necessary for "Authorization Code" grant)
- Request a token (session opening or refreshing)
- Session closing



1.1 Authentication

GET	/OAuth/Account/login	Ask for an authorization code. Returns the login window of OAuth

Parameters:

Name	Value	Description	
response_type	**code**	Authorization provider.	*mandatory*
redirect_uri	*string*	Redirection URI as registered in the OAuth server's repository.	*mandatory*
client_id	*string*	Client application as registered in the OAuth server's repository.	*mandatory*
scope	***[scopes]***	List of services allowed to the client See	*mandatory*

Responses:

Code	Description	
200	Successful	No message
404	Not found	Bad URI
400	Bad request	Send by server: bad parameters

POST	/OAuth/token	Returns token for the session

Parameters:

Name	***Value***	***Description***	
grant_type	*[grant_type]*	Authorization provider.	*mandatory*
client_id	*string*	Client application as registered in OAuth server's repository.	*mandatory*
client_secret	*string*	Secret of the client application as registered in OAuth server.	
scope	*[scopes]*	List of services allowed to the client	
redirect_uri	*string*	Redirection URI as registered in the OAuth server's repository.	
username	*string*	For grant "password": login to be sent	
password	*string*	For grant "password": password to be sent	
code	*string*	For grant "code", authorization code previously issued by the authorization server.	
refresh_token	*string*	For grant "refresh_token": Refresh token previously given by the authorization server.	

Exemple playload – format mandatory : x-www-form-urlencoded

```
{
   "grant_type": "password",
   "username": "web",
   "password": "web",
   "client_id": "SvQualifTool",
   "client_secret": "b0fd9ec2-f7d2-4984-b2c0-9325df43ed9d",
   "scope": "RealtimeData RealtimeAlarm HistoricalData GraphicalData"
}
```

Responses:

Code	***Description***	
200	Succesfull	JSON message
404	Not found	Bad uri

400	Bad request	invalid_clientId, invalid_grant, E_AccessDenied, E_TooManyUsers, E_UnknownUser, E_SvNotAccessible

Response (JSON)


```
{
   "access_token": "oY0QbrNEx6UoNmozxugKYKMYHPPuj0MSz9ZYqKJn-
3FhzPzz5A5tKldPNv2bxSMZ0fXNLJo2QGzBFY5FcPYMrbMACPzmFJoJQTqXYmF3czHF7MRxBwFBOuT2yMllqaCSh5o0WCEn50F
xNGAA82sFbYx1w0V1QsjYed372Ck3DwI7n_fRUQn3qwllzpbRxV9mz1veEOq4B0OPt5M-
RwpayFvwK83wl1PosdCupvt_BbzOzoeusxLLNPrrbSksmkl4_CiELOXuk7yKFGlKPrwL47pJ30pirIX7x1SccLqdgHb5GlqGQx
cJtpS6xpPONUbfftzFe3ZNtxCLmxXAYhBi3f--nhjuNy_LuqF_ASNQ2wULyTW4",
   "token_type": "bearer",
   "expires_in": 1199,
   "refresh_token": "b55f5935-0fe1-4a50-b9a7-6693d194501e"
}
```


1.2 Logout

GET	**/OAuth/Account/logout**	End the session

Parameters:

Only **Bearer token** in header

Responses:

Code	Description	
200	Succesfull	No message
404	Not found	Bad uri
401	Unauthorized	E_InvalidSessionId
400	Bad request	Send by server: bad parameters

1.3 Schemas

Grant_type	*Description*	*Mandatory parameters*
authorization_code	Used after LOGIN request	Code, clientID
password	Password Grant	ClientID, ClientSecret, login, password
refresh_token	Refresh token Grant	ClientID, ClientSecret, Refresh_token

Scopes	*Description*	
RealtimeData	Allow RealtimeData access	
RealtimeAlarm	Allow RealtimeAlarm access	
HistoricalData	Allow HistoricalData access	
GraphicalData	Allow GraphicalData access	

1.4 Unauthorized error and timeouts

The figure below explains how to understand and to manage "HTTP 401 UnAuthorized" errors.

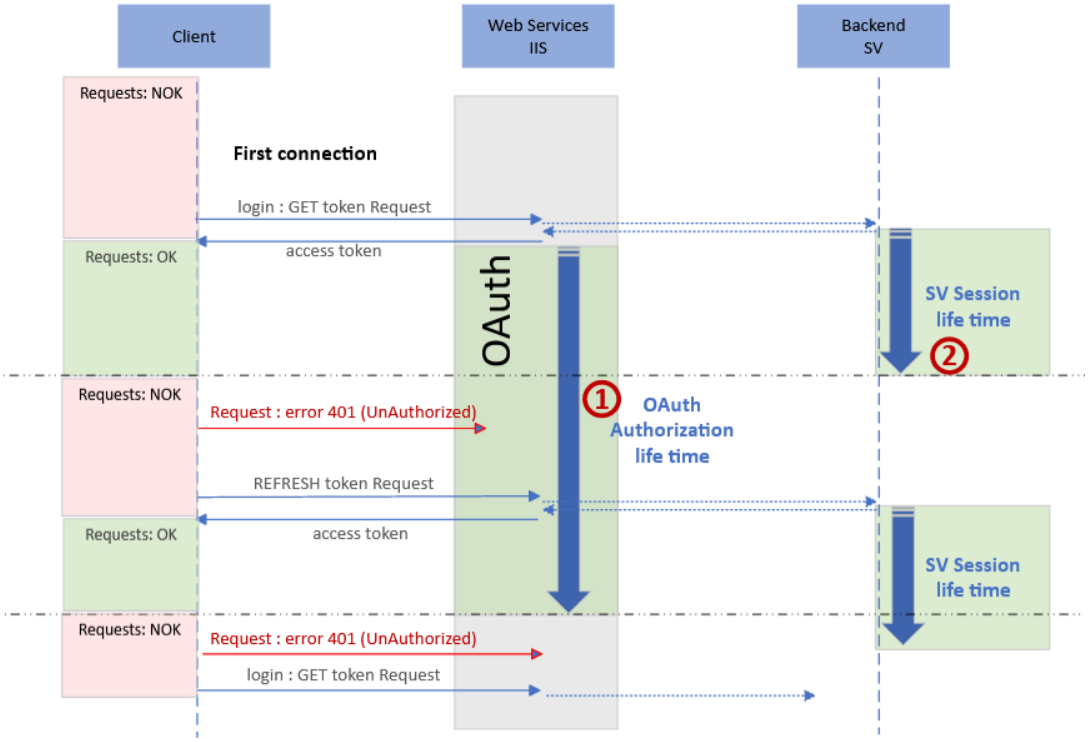


Figure 6 : Timeouts

[1] OAuth Token will expire according to an "Authorization life time". This parameter is sent with the access token message in the property "expires_in" (in seconds).

```
{
  "access_token":"oY0QbrU(...)AS4",
  "token_type":"bearer",
  "expires_in":1199,
  "refresh_token":"b55f5935(...)01e"
}
```

Expiration date can be check in the Web Deployment Console, menu "Token".

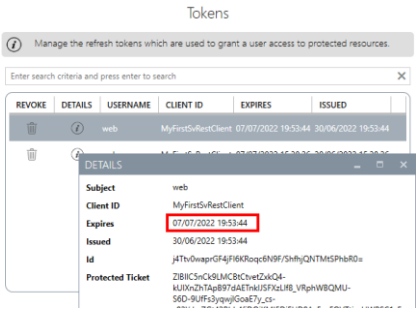


Figure 7 : OAuth Authorization timeout

---

[2] SV session will expire if SV does not receive any solicitation during a specific time, which is called session timeout.

Session timeout can be customized in SV project running on the Backend:

➔ Project setting - Web back ends

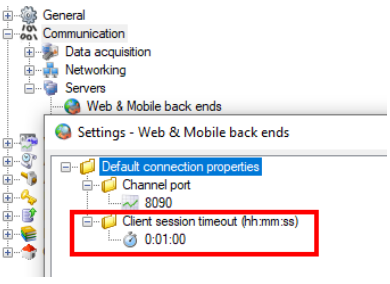


Figure 8 : SV session timeout

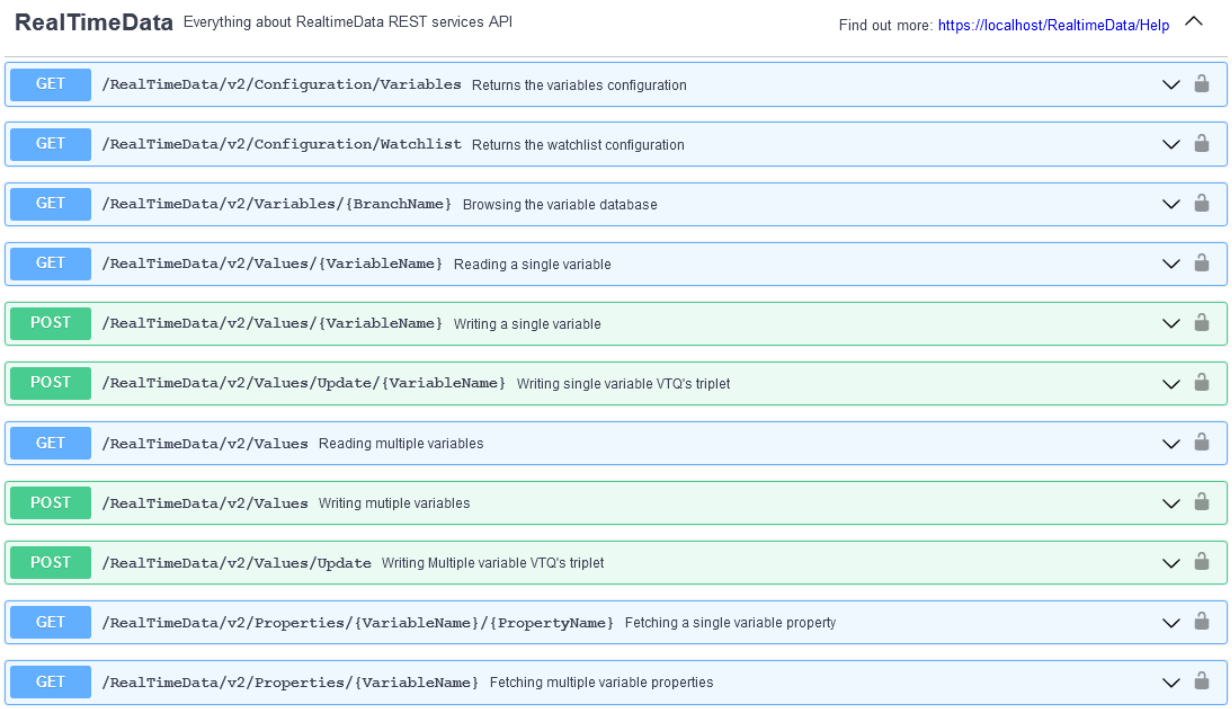
2. Real time data access service

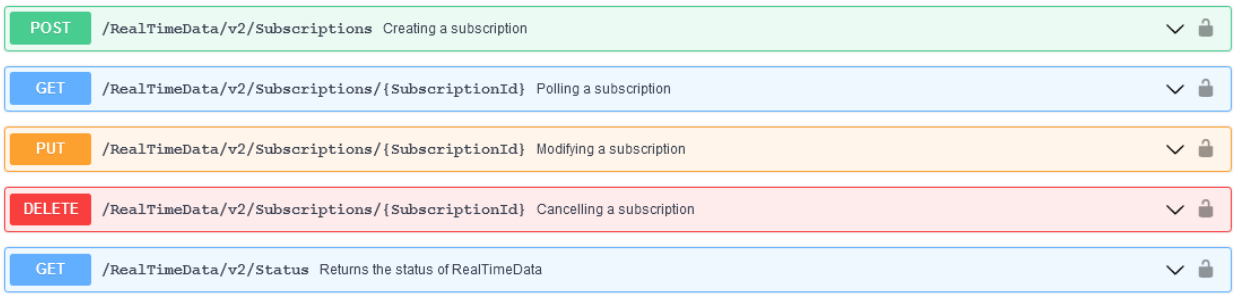
This service allows getting data in relation with SV real time variables.

Though web service ***RealtimeData***, the following operations are possible:

- Browse variables
- Read one or more variable values
- Write one or more variable values
- Set VTQs of one or more variables
- Subscribe to a variable list
- Get subscribed variable values
- Modify subscription parameters
- Unsubscribe from a variable list
- Get one or more variable properties
- Get general configuration

A *Token* must be requested prior to use this set of methods.





2.1 Common

For all methods:

- A **Bearer token** is needed for the request to be authorized (in request header). Token must be delivered by OAuth web service.
- Allowed formats for body parameters: JSON
- HTTP response table:

Code	Description	
200	Successful	No message
404	Not found	Bad uri
401	Unauthorized	Session expired (server side: E_InvalidSessionId) or token expired
400	Bad request	Bad parameters (sent by server)

2.2 Status

GET	**/RealTimeData/v2/Status**	Returns the status of RealTimeData

Parameters:

No parameter

Responses:

Response (JSON)

```
{
   "code": {
      "value": 1,
      "label": "S_Ok"
   },
   "description": ""
}
```

2.3 Browsing

GET	/RealTimeData/v2/Variables/{BranchName}	Browsing the variable database

Parameters:

Name	Value	Description
{BranchName}	*string($.../...)*	Filter by branch. Ex: *.../v2/Variables/system/hds*
Type	*array of [VariableType]*	Filter by variable type.
Depth	*integer*	Number of branches, default is 1
Size	*integer*	Number of items to be returned
Offset	*integer*	
ContinuationPoint	*string*	Receiving the next 500 items from the named variable.

Example URL

.../RealtimeData/v2/Variables/system/localhost?Depth=0&Size=2

Responses:

Response (JSON)


```
{
   "variableCollectionIterator": {
      "branches": [
         "system",
         "localhost"
      ],
      "variableType": [
         "Any"
      ],
      "VariableName": "System.LocalHost.User",
      "CollectionRecordSize": 2,
      "DepthLevel": 0,
      "FilterCriteria": ""
   },
   "variableCollections": [
      {
         "branches": [
            "system",
            "localhost"
         ],
         "variableType": "Text",
         "VariableName": "Profile",
         "IsReadOnly": true,
         "IsLeaf": true,
         "Label": ""
      },
      {
         "branches": [
            "system",
            "localhost"
         ],
         "variableType": "Text",
         "VariableName": "User",
         "IsReadOnly": true,
         "IsLeaf": true,
         "Label": ""
      }
   ]
}
```

2.4 Read variables

GET	/RealTimeData/v2/Values/{VariableName}	Reading a single variable

Parameters:

Name	Value	Description	
{VariableName}	*string($.../...)*	Variable name to read.	*mandatory*
properties	*array of [VariableProperties]*	Additional properties to be returned.	
language	*[SVLanguage]*	Language setting.	
Example URL			
.../RealtimeData/v2/Values/Engine01/P1/Temp?Properties=VariableName&Language=AlternativeLanguage&Properties=Description&Properties=TextAttr03&Properties=TextAttr04			

Responses:

Response (JSON)


```
{
   "Engine01.P1.Temp": {
      "result": {
         "code": {
            "value": 1,
            "label": "S_Ok"
         },
         "Description": null
      },
      "value": 56,
      "Timestamp": "2022-06-29T09:23:53.351",
      "quality": "Good",
      "properties": [
         "Engine01.P1.Temp",
         "",
         "Site B",
         ""
      ],
      "IsReadOnly": true,
      "QualityValue": 192,
      "alarm": null
   }
}
```


GET	/RealTimeData/v2/Values	Reading multiple variables	(beware of URL length)

Parameters:

Name	Value	Description	
variables	*array of string*	Name of the variable to be returned.	*mandatory*
properties	*array of [VariableProperties]*	Additional properties to be returned.	
language	*[SVLanguage]*	Language setting.	
Example URL			
.../RealtimeData/v2/Values/?Language=2&Variables[0]=varsets.varset001.Text&Variables[1]=varsets.varset001.Register&Properties=VariableName&Properties=Description&Properties=TextAttr03			

Responses:

Response (JSON)


```
{
   "varsets.varset001.Text": {
      "result": {
         "code": {
            "value": 1,
            "label": "S_Ok"
         },
         "Description": null
      },
      "value": "",
      "Timestamp": "2022-06-29T08:30:40.364",
      "quality": "Bad",
      "properties": [
         "varsets.varset001.Text",
         "Description 2 of Qualif_12_WST_REST.varsets.varset001.Text",
         "varsets.varset001.ExtAttr.AttrExtContent_03"
      ],
      "IsReadOnly": false,
      "QualityValue": 0,
      "alarm": null
   },
   "varsets.varset001.Register": {
      "result": {
         "code": {
            "value": 1,
            "label": "S_Ok"
         },
         "Description": null
      },
      "value": 0,
      "Timestamp": "2022-06-29T08:31:21.273",
      "quality": "Good",
      "properties": [
         "varsets.varset001.Register",
         "Description 2 of Qualif_12_WST_REST.varsets.varset001.Register",
         "varsets.varset001.ExtAttr.AttrExtContent_03"
      ],
      "IsReadOnly": false,
      "QualityValue": 192,
      "alarm": null
   }
}
```


Remark:

In HTTP, parameters of a request using GET method must be directly inserted in the URL. The used URL is then formatted as following:
https://url?param1=value1&param2=value2
The consequence is that reading many variables with **/RealTimeData/v2/Values** adress can generate a very long URL, which can exceed the maximum length allowed by clients (browsers, libraries…).
To read a large number of variables, it is better to use **/RealTimeData/v2/BulkRead** function, which use POST method.

POST	/RealTimeData/v2/BulkRead	Reading multiple variables (without URI length limitation)

Parameters:

Name	Value	Description	
variables	*array of string*	Name of the variable to be returned	*mandatory*
properties	*array of [VariableProperties]*	Additionnal properties to be returned.	
language	*[SVLanguage]*	Language setting	

Exemple playload (JSON)

```
{
   "Variables": [
      "varsets.varset001.Text",
      "varsets.varset001.Register"
   ],
   "Properties": [
      "VariableName",
      "Description",
      "TextAttr03"
   ]
}
```

Responses:

Response (JSON)

```
{
   "varsets.varset001.Text": {
      "result": {
         "code": {
            "value": 1,
            "label": "S_Ok"
         },
         "Description": null
      },
      "value": "",
      "Timestamp": "2022-06-29T08:30:40.364",
      "quality": "Bad",
      "properties": [
         "varsets.varset001.Text",
         "Description 2 of Qualif_12_WST_REST.varsets.varset001.Text",
         "varsets.varset001.ExtAttr.AttrExtContent_03"
      ],
      "IsReadOnly": false,
      "QualityValue": 0,
      "alarm": null
   },
   "varsets.varset001.Register": {
      "result": {
         "code": {
            "value": 1,
            "label": "S_Ok"
         },
         "Description": null
      },
      "value": 0,
      "Timestamp": "2022-06-29T08:31:21.273",
      "quality": "Good",
      "properties": [
         "varsets.varset001.Register",
         "Description 2 of Qualif_12_WST_REST.varsets.varset001.Register",
         "varsets.varset001.ExtAttr.AttrExtContent_03"
      ],
```

```
        "IsReadOnly": false,
        "QualityValue": 192,
        "alarm": null
    }
}
```


2.5 Read properties

GET	/RealTimeData/v2/Properties/{VariableName}	Fetching variable properties

Parameters:

Name	Value	Description	
{VariableName}	*string($.../...)*	Variable name to read.	*mandatory*
Properties	*array of [VariableProperties]*	Properties to be returned.	*mandatory*
Example URL			
.../RealtimeData/v2/Properties/varsets/varset001/Register?Properties=VariableName&Properties=Description&Properties=TimestampType&Properties=TextAttr03&Properties=MinValue&Properties=MaxValue&Properties=Unit			

Responses:

Response (JSON)


```
[
    {
        "result": {
            "code": {
                "value": 1,
                "label": "S_Ok"
            },
            "Description": null
        },
        "properties": [
            "varsets.varset001.Register",
            "Description 1 of Qualif_12_WST_REST.varsets.varset001.Register",
            "SV",
            "varsets.varset001.ExtAttr.AttrExtContent_03",
            0,
            65535,
            "reg unit"
        ]
    }
]
```

2.6 Write variables

POST	/RealTimeData/v2/Values/{VariableName}	Writing a single variable

Parameters:

Name	*Value*	*Description*	
{VariableName}	*string($.../...)*	Variable name to write.	*mandatory*
Value	*string*	Value to write	*mandatory*

Example URL

.../RealtimeData/v2/Values/varsets/varset001/Register

Exemple playload (JSON)

```
{
   "value": "22"
}
```

Responses:

Response (JSON)

```
[
   {
      "code": {
         "value": 1,
         "label": "S_Ok"
      },
      "description": null
   }
]
```

POST	/RealTimeData/v2/Values	Writing mutiple variables

Parameters:

Name	*Value*	*Description*	
variables	*array of string*	Name of the variable to be returned	*mandatory*
values	*array of string*	Value to write in each variable	*mandatory*
CustomUserName	*string*	Alternative username to be used by SV	

Exemple playload (JSON)

```
{
   "customUserName": "Ralph",
   "variables": [
      "varsets.varset001.Register",
      "varsets.varset001.Text"
   ],
   "values": [
      "77.7",
      "Salut"
   ]
}
```

Responses:

Response (JSON)

```
[
   {
      "code": {
         "value": 1,
         "label": "S_Ok"
```

```
        },
        "description": null
    },
    {
        "code": {
            "value": 1,
            "label": "S_Ok"
        },
        "description": null
    }
]
```


POST	/RealTimeData/v2/Values/Update/{VariableName}	Writing single variable VTQ's triplet

Parameters:

Name	***Value***	***Description***	
{VariableName}	*string($.../...)*	Variable name to write.	*mandatory*
Value	*[UpdateValue]*	Value to write	*mandatory*
CustomUserName	*string*	Alternative username to be used by SV	
Example URL			
.../RealtimeData/v2/Values/varsets/varset001/Register			
Exemple playload (JSON)			


```
{
    "Value": {
        "Timestamp": "2018-07-02T13:42:09.0061806+02:00",
        "Quality": "QualityBadWatchdogFailure",
        "value": "1.32"
    }
}
```


Responses:

Response (JSON)


```
[
    {
        "code": {
            "value": 1,
            "label": "S_Ok"
        },
        "description": null
    }
]
```


POST	/RealTimeData/v2/Values/Update	Writing Multiple variable VTQ's triplet

Parameters:

Name	***Value***	***Description***	
values	*array of [UpdateValue]*	Value to write in each variable	*mandatory*
Exemple playload (JSON)			


```
{
    "Values": [
        {
            "name": "varsets.varset002.Register",
            "Timestamp": "2018-06-02T13:42:09.0061806+02:00",
```

```
            "Quality": "0",
            "value": "617"
        },
        {
            "name": "varsets.varset002.Register",
            "Timestamp": "2018-06-02T12:42:09.0061806+02:00",
            "Quality": "0",
            "value": "244"
        }
    ]
}
```


Responses:

Response (JSON)


```
[
    {
        "code": {
            "value": 1,
            "label": "S_Ok"
        },
        "description": null
    },
    {
        "code": {
            "value": 1,
            "label": "S_Ok"
        },
        "description": null
    }
]
```

2.7 Subscription

POST	/RealTimeData/v2/Subscriptions	Creating a subscription

Parameters:

Name	***Value***	***Description***	
properties	*array of [VariableRequestParameters]*	Subscription parameters	*mandatory*
EnableBuffering	*string* ("true" *or* "false")	Ask server to retain every value between two requests	

Exemple playload (JSON)

```
{
   "EnableBuffering": "false",
   "Properties": {
      "projectLanguage": "1",
      "variableNames": [
         "varsets.varset001.Register",
         "varsets.varset001.Ala"
      ],
      "variableProperties": [
         "Description",
         "TimestampType",
         "varName"
      ]
   }
}
```

Responses:

Response (JSON)

```
{
   "subscriptionId": "961274e5-670c-47a0-99f5-f8371d4646bc--16--",
   "subscriptionResults": [
      {
         "code": {
            "value": 1,
            "label": "S_Ok"
         },
         "description": null
      },
      {
         "code": {
            "value": 1,
            "label": "S_Ok"
         },
         "description": null
      }
   ]
}
```

GET	/RealTimeData/v2/Subscriptions/{SubscriptionId}	Polling a subscription

Parameters:

Name	**Value**	**Description**	
{SubscriptionId}	*string*	Subscription Id	*mandatory*

Example URL

.../RealtimeData/v2/Subscriptions/961274e5-670c-47a0-99f5-f8371d4646bc--16--/

Responses:

Response (JSON)


```
[
   {
      "result": {
         "code": {
            "value": 1,
            "label": "S_Ok"
         },
         "description": null
      },
      "index": 0,
      "values": [
         {
            "result": {
               "code": {
                  "value": 1,
                  "label": "S_Ok"
               },
               "description": null
            },
            "value": 1.32,
            "timestamp": "2018-07-02T13:42:09.006",
            "quality": "Bad",
            "properties": [
               "Description 1 of Qualif_12_WST_REST.varsets.varset001.Register",
               "SV"
            ],
            "isReadOnly": false,
            "qualityValue": 0,
            "alarm": null
         }
      ]
   }
]
```


PUT	**/RealTimeData/v2/Subscriptions/{SubscriptionId}**	Modifying a subscription

Parameters:

Name	Value	Description	
{SubscriptionId}	*string*	Subscription Id	*mandatory*
properties	*array of [VariableRequestParameters]*	Subscription parameters	*mandatory*
EnableBuffering	*string* ("true" *or* "false")	Ask server to retain every value between two requests	

Example URL

.../RealtimeData/v2/Subscriptions/961274e5-670c-47a0-99f5-f8371d4646bc--16--/

Exemple playload (JSON)


```
{
   "EnableBuffering": "false",
   "Properties": {
      "projectLanguage": "1",
      "variableNames": [
         "varsets.varset001.Register",
```

```
            "varsets.varset001.Text",
            "varsets.varset001.Bit",
            "varsets.varset001.Ala"
        ],
        "variableProperties": [
            "Description",
            "TimestampType",
            "varName"
        ]
    }
}
```

Responses:

Response (JSON)


```
{
    "subscriptionId": "0d359493-5e26-4783-ac4c-d9d6bfabb170--16--",
    "subscriptionResults": [
        {
            "code": {
                "value": 1,
                "label": "S_Ok"
            },
            "description": null
        },
        {
            "code": {
                "value": 1,
                "label": "S_Ok"
            },
            "description": null
        },
        {
            "code": {
                "value": 1,
                "label": "S_Ok"
            },
            "description": null
        },
        {
            "code": {
                "value": 1,
                "label": "S_Ok"
            },
            "description": null
        }
    ]
}
```


DELETE	/RealTimeData/v2/Subscriptions/{SubscriptionId}	Cancelling a subscription

Parameters:

Name	Value	Description	
{SubscriptionId}	*string*	Subscription Id	*mandatory*
Example URL			
.../RealtimeData/v2/Subscriptions/961274e5-670c-47a0-99f5-f8371d4646bc--16--/			

Responses:

Response (JSON)


```
{
   "code": {
      "value": 1,
      "label": "S_Ok"
   },
   "description": null
}
```


2.8 Schemas

SVLanguage

Name	*Value*	*Description*
ContextDefault	0	
BaseLanguage	1	
AlternativeLanguage	2	

VariableType

Name	*Value*	*Description*
Any	0	
Bit	1	
Alarm	2	
Register	3	
Text	4	
Branch	5	

VariableProperty

Name	*Value*	*Description*
VariableName	0	
Description	1	
TimestampType	2	
AssocLabel	3	
TextAttr01 → TextAttr16	4 → 19	
DeferredTextAttr03 → DeferredTextAttr16	20 → 33	
BinAttr	34	
MinValue	35	
MaxValue	36	
MinControlValue	37	
MaxControlValue	38	
MinVariableName	39	
MaxVariableName	40	
MinControlVariableName	41	
MaxControlVariableName	42	
Unit	43	
Format	44	
Anim0AssocLabel	45	
Anim1AssocLabel	46	

Cmd0AssocLabel	47	
Cmd1AssocLabel	48	
Log0AssocLabel	49	
Log1AssocLabel	50	
FormattedValue	51	
AlarmLevel	52	
Maintenance	53	
VariableType	54	

UpdateValue

Name	*Value*	*Description*
Name	*string*	
Value	string	
Quality	[UpdateValueQuality]	
Timestamp	string	

UpdateValueQuality

Name	*Value*	*Description*
QualityGood	0	
QualityBad	1	
QualityBadCommunicationFailure	2	
QualityBadBadFormattingValue	3	
QualityBadWatchdogFailure	4	

VariableRequestParameters

Name	*Value*	*Description*
variableNames	*array of string*	
variableProperties	*array of [VariableProperty]*	
projectLanguage	*[SVLanguage]*	
Context	*string*	

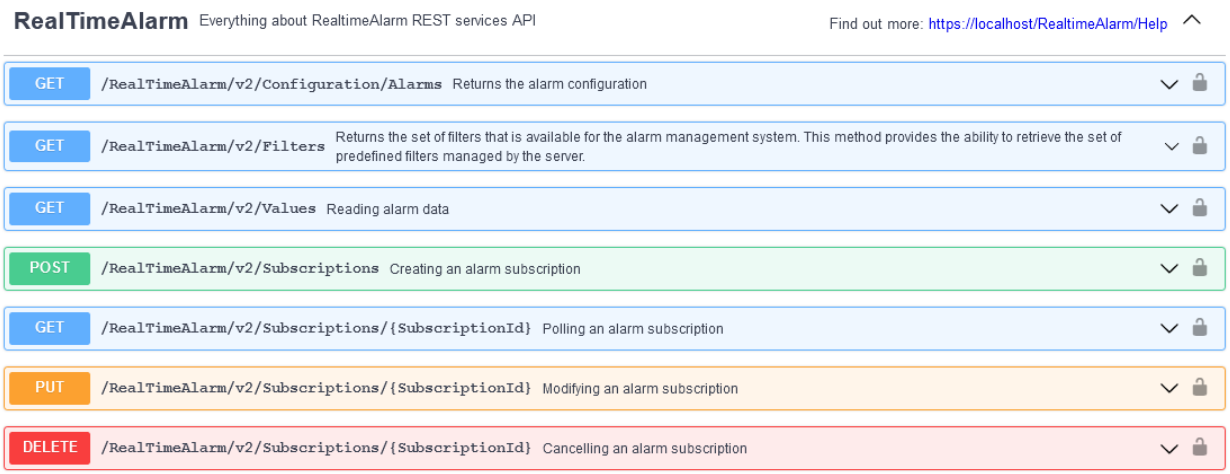
3. Alarm list access service

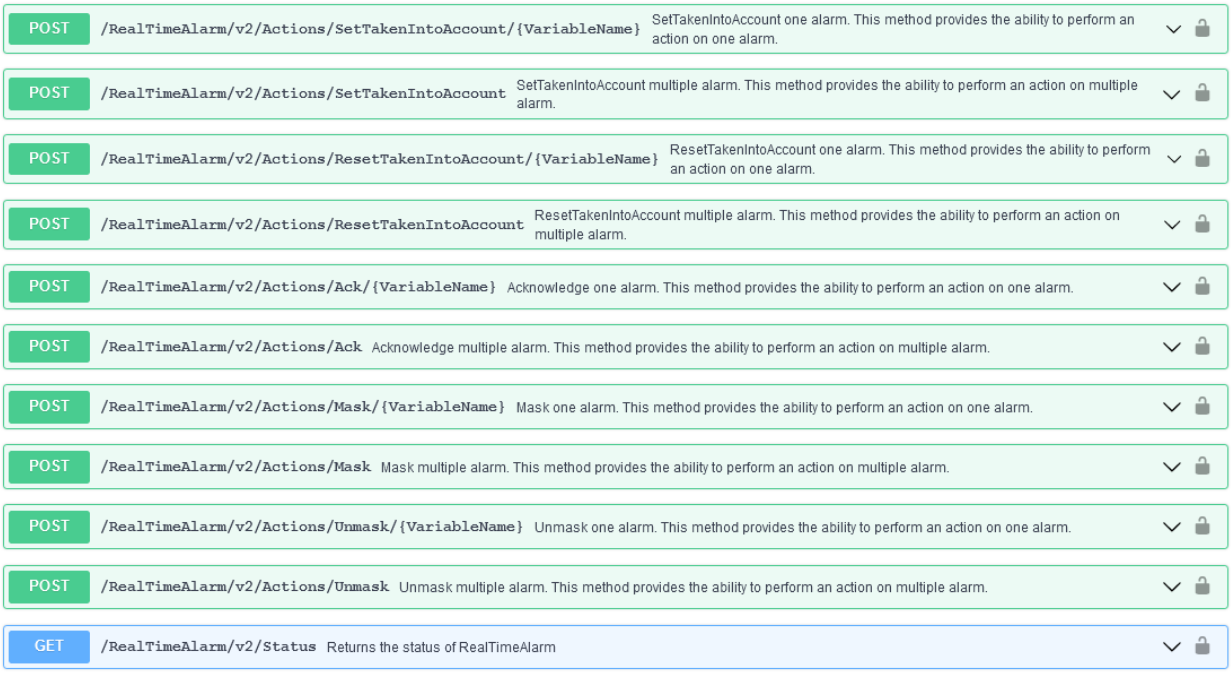
This service allows getting alarm data in relation with the SV alarm manager.

Through web service ***RealtimeAlarm***, the following operations are possible:

- Subscribe to an alarm filter
- Set subscription parameters
- Request alarm data for an alarm subscription
- Unsubscribe from an alarm list
- Get alarm associated labels
- Get filter names list
- Execute user actions (acknowledgement, masking, unmasking …)

A *Token* must be requested prior to using this set of methods.





3.1 Common

For all methods:

- A **Bearer token** is needed for the request to be authorized (in request header). Token must be delivered by OAuth web service.
- Allowed formats for body parameters: JSON
- HTTP response table:

Code	Description	
200	Successful	No message
404	Not found	Bad uri
401	Unauthorized	Session expired (server side: E_InvalidSessionId) or token expired
400	Bad request	Bad parameters (sent by server)

3.2 Status

GET	**/RealTimeAlarm/v2/Status**	Returns the status of RealTimeAlarm

Parameters:

No parameter

Responses:

Response (JSON)

```
{
   "code": {
      "value": 1,
      "label": "S_Ok"
   },
   "description": ""
}
```

3.3 Read

GET	/RealTimeAlarm/v2/Values	Reading alarm data

Parameters:

Name	Value	Description	
Events	*array of [AlarmSupportedEventCode]*	Event codes for alarm list items to be returned by the request.	*mandatory*
Language	*[SVLanguage]*	SV language for the request. If not set, use of session language.	
MinLevel	*integer*	Minimum alarm level	
MaxLevel	*integer*	Maximum alarm level	
Size	*integer*	Maximum Number of items to retrieve	
Properties	*array of [AlarmProperty]*	Additional properties to be provided for each returned alarm list item.	
PriorityFilter	*[PriorityFilter]*	Defines how to interpret the minimum level and maximum level properties	
Filters	*array of string*	Additional filter criteria to be applied to the request	
LogicalOperator	*boolean*	Operator applied to the filter criteria. True = logical AND, False = logical OR	

Example URL

.../RealtimeAlarm/v2/Values?Language=1&Size=50&MinLevel=0&MaxLevel=4&PriorityFilter=All&Events[0]=AlarmUnavailable&Properties[0]=0&Properties[1]=1&Properties[2]=2

Responses:

Response (JSON)


```
{
   "values": [
      {
         "removedFromList": false,
         "timestamp": "2022-06-29T08:30:40.363",
         "eventCode": "AlarmUnavailable",
         "properties": [
            "varsets.varset001.Ala",
            "la_ns",
            "Unavailable"
         ],
         "eventValue": 16
      },
      {
         "removedFromList": false,
         "timestamp": "2022-06-29T08:30:40.364",
         "eventCode": "AlarmUnavailable",
         "properties": [
            "varsets.varset002.Ala",
            "la_ns",
            "Unavailable"
         ],
         "eventValue": 16
      },
      {
         "removedFromList": false,
         "timestamp": "2022-06-29T08:30:40.393",
         "eventCode": "AlarmUnavailable",
```

```
            "properties": [
               "br1.br2.br3.br4.br5.br6.br7.br8.br9.br10.AlarmEventCodes.4",
               "la_ns",
               "Unavailable"
            ],
            "eventValue": 16
         }
      ],
      "maxNumberExceeded": false
}
```


3.4 Actions

POST **/RealTimeAlarm/v2/Actions**

/SetTakenIntoAccount/{VariableName}	SetTakenIntoAccount one alarm
/ResetTakenIntoAccount/{VariableName}	ResetTakenIntoAccount one alarm
/Ack/{VariableName}	Acknowledge one alarm
/Mask/{VariableName}	Mask one alarm
/Unmask/{VariableName}	Unmask one alarm

Parameters:

Name	***Value***	***Description***	
{VariableName}	*string($.../...)*	Alarm to be set	*mandatory*
Properties	*array of [ExecuteUserActionParameters]*	Additional properties	

Example URL

.../RealtimeAlarm/v2/Actions/SetTakenIntoAccount/varsets/varset004/AlarmEventCodes/0

Exemple playload (JSON)


```
{
   "Properties": {
      "CustomUserName": "Ralph"
   }
}
```


Responses:

Response (JSON)


```
[
   {
      "code": {
         "value": 1,
         "label": "S_Ok"
      },
      "description": null
   }
]
```

POST	/RealTimeAlarm/v2/Actions	
	/SetTakenIntoAccount	SetTakenIntoAccount multiple alarm
	/ResetTakenIntoAccount	ResetTakenIntoAccount multiple alarm
	/Ack	Acknowledge multiple alarm
	/Mask	Mask multiple alarm
	/Unmask	Unmask multiple alarm

Parameters:

Name	***Value***	***Description***	
Alarms	*array of string*	Alarms to be set	*mandatory*
Properties	*array of [ExecuteUserActionParameters]*	Additional properties	

Exemple playload (JSON)

```
{
   "Alarms": [
      "varsets.varset004.AlarmEventCodes.1",
      "varsets.varset004.AlarmEventCodes.2",
      "varsets.varset004.AlarmEventCodes.3",
      "varsets.varset004.AlarmEventCodes.4"
   ]
}
```

Responses:

Response (JSON)

```
[
   {
      "code": {
         "value": 1,
         "label": "S_Ok"
      },
      "description": null
   },
   {
      "code": {
         "value": 1,
         "label": "S_Ok"
      },
      "description": null
   },
   {
      "code": {
         "value": 1,
         "label": "S_Ok"
      },
      "description": null
   },
   {
      "code": {
         "value": 1,
         "label": "S_Ok"
      },
      "description": null
   }
]
```

3.5 Subscription

POST	/RealTimeAlarm/v2/Subscriptions	Creating a subscription

Parameters:

Name	*Value*	*Description*	
properties	*array of [AlarmRequestParameters]*	Subscription parameters	*mandatory*

Exemple playload (JSON)

```
{
   "Properties": {
      "MinLevel": "0",
      "MaxLevel": "3",
      "ElementMaxNumber": "10",
      "eventCodes": [
         "AlarmOnNotAck",
         "AlarmOnAck"
      ],
      "properties": "VariableName"
   }
}
```

Responses:

Response (TEXT)

1e74e4b6-b6fe-4d7d-9264-5edd9813d106--14--

GET	/RealTimeAlarm/v2/Subscriptions/{SubscriptionId}	Polling a subscription

Parameters:

Name	Value	Description	
{SubscriptionId}	*string*	Subscription Id	*mandatory*

Example URL

.../RealtimeAlarm/v2/Subscriptions/1e74e4b6-b6fe-4d7d-9264-5edd9813d106--14--/

Responses:

Response (JSON)

```
{
   "values": [
      {
         "removedFromList": false,
         "timestamp": "2022-06-29T08:30:45.453",
         "eventCode": "AlarmOnAck",
         "properties": null,
         "eventValue": 2
      }
   ],
   "maxNumberExceeded": false
}
```

PUT	/RealTimeAlarm/v2/Subscriptions/{SubscriptionId}	Modifying a subscription

Parameters:

Name	Value	Description	
{SubscriptionId}	*string*	Subscription Id	*mandatory*
properties	*array of [AlarmRequestParameters]*	Subscription parameters	*mandatory*

Example URL
.../RealtimeData/v2/Subscriptions/961274e5-670c-47a0-99f5-f8371d4646bc--16--/

Exemple playload (JSON)

```
{
   "Properties": {
      "projectLanguage": "2",
      "ElementMaxNumber": "5",
      "MinLevel": "0",
      "MaxLevel": "29",
      "LogicalOperator": "0",
      "Filters": [
         "[Attribute03]='true'",
         "[Attribute04] = 'true'"
      ],
      "eventCodes": [
         "0",
         "1",
         "2"
      ],
      "Properties": [
         "0",
         "1"
      ],
      "Domain": "dom_P0",
      "Nature": "nat_P0"
   }
}
```

Responses:

Response (TEXT)

```
1e74e4b6-b6fe-4d7d-9264-5edd9813d106--14--
```

DELETE **/RealTimeAlarm/v2/Subscriptions/{SubscriptionId}** Cancelling a subscription

Parameters:

Name	Value	Description	
{SubscriptionId}	*string*	Subscription Id	*mandatory*

Example URL

.../RealtimeData/v2/Subscriptions/961274e5-670c-47a0-99f5-f8371d4646bc--16--/

Responses:

Response (JSON)

```
[
   {
      "code": {
         "value": 1,
         "label": "S_Ok"
      },
      "description": null
   }
]
```

3.6 Schemas

SVLanguage

Name	*Value*	*Description*
ContextDefault	0	
BaseLanguage	1	
AlternativeLanguage	2	

PriorityFilter

Name	*Value*	*Description*
All	0	
Minimum	1	
Maximum	2	

AlarmProperty

Name	*Value*	*Description*
VariableName	0	
AssocLabel	1	
StandardLabel	2	
UserName	3	
Station	4	
AlarmLevel	5	
Informations	6	
LogInformations	7	
TimestampType	8	
Description	9	
TextAttr01 → TextAttr16	10 → 25	
DeferredTextAttr03 → DeferredTextAttr16	26 → 39	
BinAttr	40	

AlarmSupportedEventCode

Name	*Value*	*Description*
AlarmOnNotAck	0	
AlarmOffNotAck	1	
AlarmOnAck	2	
AlarmOffAck	3	
AlarmUnavailable	4	
AlarmNotAccessible	5	
AlarmInhibited	6	
AlarmProgramMasked	7	
AlarmVariableMasked	8	
AlarmUserMasked	9	
AlarmExpressionMasked	10	
AlarmTakenIntoAccount	11	

AlarmRequestParameters

Name	*Value*	*Description*
projectLanguage	*[SVLanguage]*	
ElementMaxNumber	*integer*	

MinLevel	*integer*	
MaxLevel	*integer*	
eventCodes	*array of [AlarmSupportedEventCode]*	
priorityFilter	*[PriorityFilter]*	
properties	*array of [AlarmProperty]*	
filters	*array of string*	
LogicalOperator	*string*	
Domain	*string*	
Nature	*string*	
Format	*string*	
Context	*string*	
Token	*string*	
ServerId	*string*	
Sound	*string*	

ExecuteUserActionParameters

Name	*Value*	*Description*
CustomUserName	string	

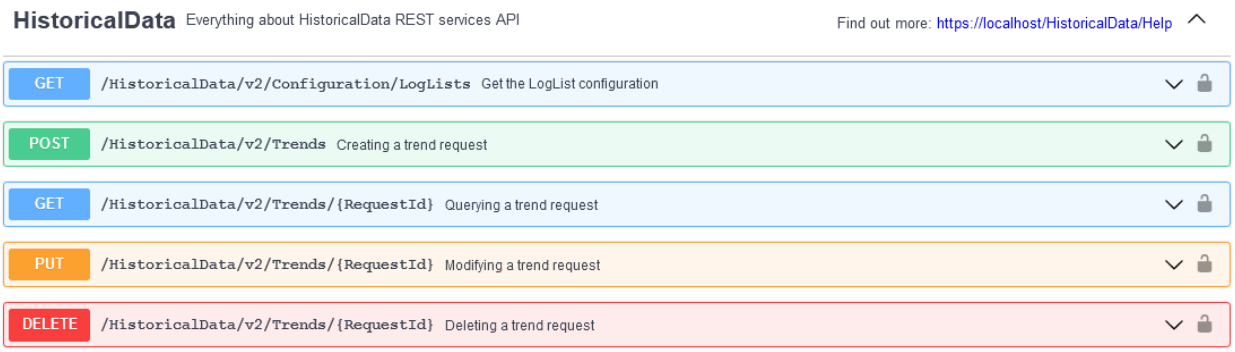
4. Historical data access service

This service allows getting trend data in relation with the SV historical manager.

Though web service ***HistoricalData***, the following operations are possible:

- Create a trend request
- Request archived trend data for a trend request
- Modify a trend request
- Delete a trend request

A *Token* must be requested prior to using this set of services.

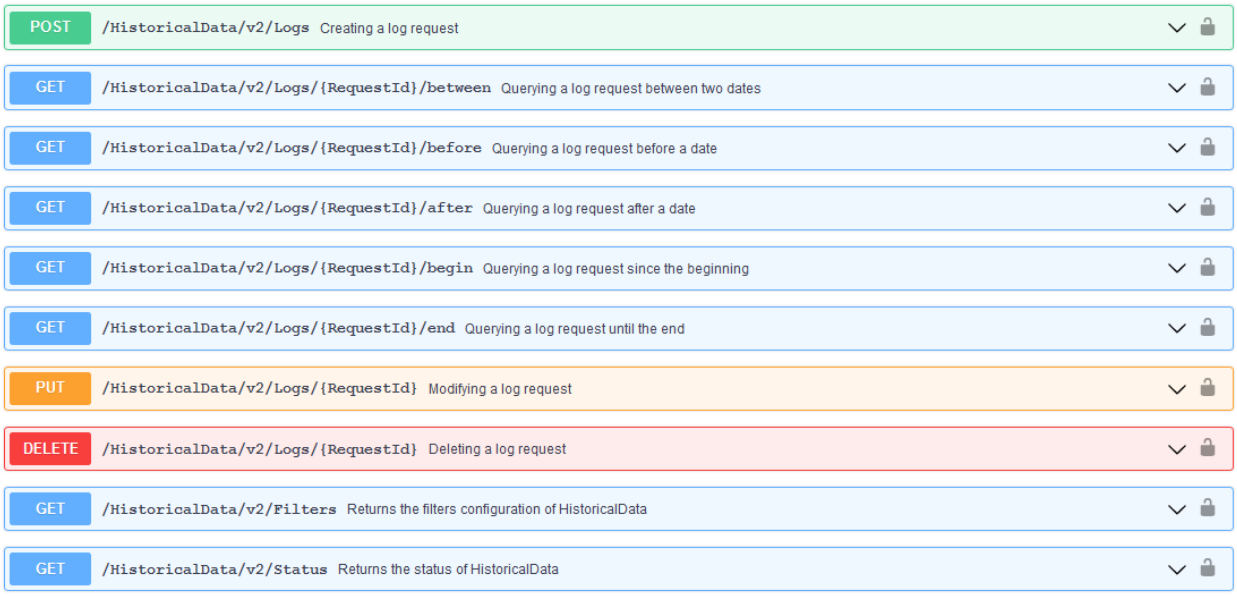


This service also allows getting archived events in relation with the SV historical manager. Those archived data are also known as logged events.

The following operations are possible:

- Create a log request
- Request logged data for a log request
- Modify a log request
- Delete a log request
- Get log list names
- Get filter names

A *Token* must be requested prior to using this set of services.



4.1 Common

For all methods:

- An authentication **Bearer token** is needed for the request to be authorized (in request header). Token must be delivered by OAuth web service.
- Allowed formats for body parameters: JSON or XML
- HTTP response table:

Code	Description	
200	Successful	No message
404	Not found	Bad uri
401	Unauthorized	Session expired (server side: E_InvalidSessionId) or token expired
400	Bad request	Bad parameters (sent by server)

4.2 Status

GET	**/HistoricalData/v2/Status**	Returns the status of HistoricalData

Parameters:

No parameter

Responses:

Response (JSON)

```
[
   {
      "code": {
         "value": 1,
         "label": "S_Ok"
      },
      "description": null
   }
]
```

4.3 Configuration

GET	/HistoricalData/v2/Configuration/LogLists	Get the LogList configuration

Parameters:

No parameter

Responses:

Response (JSON)


```
{
   "loglist": [
      {
         "name": "Main",
         "minlevel": 0,
         "maxlevel": 29,
         "maxelement": 50,
         "allowfiltering": true,
         "alarmEvents": {
            "eventCodes": {
               "eventCode": [
                  "AlarmOnNotAck",
                  "AlarmOffNotAck",
                  "AlarmOnAck",
                  "AlarmOffAck",
                  "AlarmNotAccessible"
               ]
            }
         },
         "bitChanges": {
            "eventCodes": {
               "eventCode": [
                  "BitChangeTo0",
                  "BitChangeTo1"
               ]
            }
         },
         "userActions": {
            "eventCodes": {
               "eventCode": [
                  "UserActionSendProgram",
                  "UserActionLogonLogoff"
               ]
            }
         }
      }
   ]
}
```


GET	/HistoricalData/v2/Filters	Returns the filters configuration of HistoricalData.

Parameters:

No parameter

Responses:

Response (JSON)


```
[
   {
      "name": "test_domain",
      "description": "test_domain",
      "expression": "[Attribute01]='test_domain'"
   },
   {
      "name": "nat_ala",
      "description": "nat_ala",
      "expression": "[Attribute02]='nat_ala'"
   },
   {
      "name": "nat_reg",
      "description": "nat_reg",
      "expression": "[Attribute02]='nat_reg'"
   },
   {
      "name": "Dom_P0",
      "description": "Dom_P0",
      "expression": "[Attribute01]='Dom_P0'"
   }
]
```

4.4 Trends

POST	/HistoricalData/v2/Trends	Creating a trend request

Parameters:

Name	***Value***	***Description***	
VariableName	*string*	Name of the variable to be requested	*mandatory*
ElementMaxNumber	*integer*		
aggregateFunction	*integer*	*0 = Raw, 1 = WindowPixelSize*	
AggregateParam1	*integer*		
properties	*array of [TrendProperty]*		
projectLanguage	*[SVLanguage]*		
Context	*string*		
IncludeStartBound	*boolean*		
IncludeEndBound	*boolean*		

Exemple playload (JSON)

```
{
   "VariableName": "varsets.varset001.Chrono",
   "elementMaxNumber": "100",
   "properties": [
      "VariableName",
      "Description",
      "StandardLabel"
   ]
}
```

Responses:

Response (TEXT)
0b2bdba0-1fb3-4e17-aa23-f1d419c6ab10--12--

GET	/HistoricalData/v2/Trends/{RequestId}	Querying a trend request

Parameters:

Name	**Value**	**Description**	
{RequestId}	*string*	Request Id	*mandatory*
Start	*string*	Starting date	*mandatory*
End	*string*	Ending date	*mandatory*

Example URL

.../HistoricalData/v2/Trends/0b20-1fb3-4e17-aa23-f1d419c6ab10--12--/?Start=2018-06-28 15:20:00&End=2018-07-28 16:30:00

Responses:

Response (JSON)

```
{
   "values": [
      {
         "value": 456.321,
         "timestamp": "2018-07-05T16:41:17.282",
         "quality": "Good",
         "qualityValue": 0,
         "properties": null
```

```
        },
        {
            "value": 40,
            "timestamp": "2018-07-06T08:15:54.411",
            "quality": "Good",
            "qualityValue": 0,
            "properties": null
        }
    ],
    "maxNumberExceeded": false
}
```


PUT	/HistoricalData/v2/Trends/{RequestId}	Modifying a trend reques

Parameters:

Name	Value	Description	
{RequestId}	*string*	Request Id	*mandatory*
VariableName	*string*	Name of the variable to be requested	*mandatory*
ElementMaxNumber	*integer*		
aggregateFunction	*integer*	*0 = Raw, 1 = WindowPixelSize*	
AggregateParam1	*integer*		
properties	*array of [TrendProperty]*		
projectLanguage	*[SVLanguage]*		
Context	*string*		
IncludeStartBound	*boolean*		
IncludeEndBound	*boolean*		

Example URL

.../HistoricalData/v2/Trends/0b2bdba0-1fb3-4e17-aa23-f1d419c6ab10--12--/

Exemple playload (JSON)


```
{
    "VariableName": "varsets.varset007.Chrono",
    "ElementMaxNumber": "1000",
    "Context": "\"Qualification\"",
    "includeStartBound": "true",
    "includeEndBound": "true",
    "properties": [
        "VariableName"
    ]
}
```


Responses:

Response (JSON)


```
{
    "code": {
        "value": 1,
        "label": "S_Ok"
    },
    "Description": null
}
```

DELETE	/HistoricalData/v2/Trends/{RequestId}	Deleting a trend request

Parameters:

Name	Value	Description	
{RequestId}	*string*	Request Id	*mandatory*
Example URL			
.../HistoricalData/v2/Trends/0b2bdba0-1fb3-4e17-aa23-f1d419c6ab10--12--/			

Responses:

Response (JSON)

```
{
   "code": {
      "value": 1,
      "label": "S_Ok"
   },
   "Description": null
}
```

4.5 Logs

POST	/HistoricalData/v2/Logs	Creating a log request

Parameters:

Name	***Value***	***Description***	
LogListName	*string*	Name of the requested log list	*mandatory*
ElementMaxNumber	*integer*		
projectLanguage	*[SVLanguage]*		
eventCodes	*array of [LogSupportedEventCode]*		
MinLevel	*integer*		
MaxLevel	*integer*		
LogicalOperator	*boolean*		
properties	*array of [LogProperty]*		
filters	*array of string*		
AutoRefresh	*boolean*		
RefreshPeriod	*integer*		
Format	*string*		
Domain	*string*		
Nature	*string*		
Context	*string*		
LogIndex	*[LogIndex]*		

Exemple playload (JSON)

```
{
   "LogListName": "Actions",
   "projectLanguage": "ContextDefault",
   "ElementMaxNumber": "100",
   "properties": [
      "VarName",
      "Description"
   ],
   "eventCodes": [
      "BitChangeTo0",
      "BitChangeTo1",
      "AlarmOnNotAck",
      "AlarmOffNotAck",
      "AlarmOnAck",
      "AlarmOffAck",
      "UserActionAlarmAcknowledgement",
      "UserActionAlarmMaskUnmask",
      "UserActionSendCommand",
      "UserActionLogonLogoff"
   ]
}
```

Responses:

Response (TEXT)
0d29e467-6084-4e4f-9504-eb3541dafdee--9--

GET	/HistoricalData/v2/Logs/{RequestId}	
	/between	Querying a log request between two dates
	/before	Querying a log request before a date
	/after	Querying a log request after a date
	/begin	Querying a log request since the beginning
	/end	Querying a log request until the end

Parameters:

Name	Value	Description	
{RequestId}	*string*	Request Id	*mandatory*
Start		Starting date	*mandatory for between, after*
End		Ending date	*mandatory for between, before*
Example URL			
.../HistoricalData/v2/Logs/084-4e4f-9504-eb31dafdee--9--/between/?Start=2018-06-28 16:20:00&End=2018-07-09 17:20:00			

Responses:

Response (JSON)


```
{
   "values": [
      {
         "timestamp": "2018-07-06T10:26:23.359",
         "eventCode": "UserActionLogonLogoff",
         "corrupted": false,
         "eventValue": 256,
         "properties": [
            ""
         ]
      },
      {
         "timestamp": "2018-07-06T10:53:12.718",
         "eventCode": "UserActionLogonLogoff",
         "corrupted": false,
         "eventValue": 128,
         "properties": [
            ""
         ]
      }
   ],
   "maxNumberExceeded": false
}
```


PUT	/HistoricalData/v2/Logs/{RequestId}	Modifying a log request

Parameters:

Name	Value	Description	
{RequestId}	*string*	Request Id	*mandatory*
LogListName	*string*	Name of the requested log list	*mandatory*
ElementMaxNumber	*integer*		
projectLanguage	*[SVLanguage]*		

eventCodes	*array of [LogSupportedEventCode]*		
MinLevel	*integer*		
MaxLevel	*integer*		
LogicalOperator	*boolean*		
properties	*array of [LogProperty]*		
filters	*array of string*		
AutoRefresh	*boolean*		
RefreshPeriod	*integer*		
Format	*string*		
Domain	*string*		
Nature	*string*		
Context	*string*		
LogIndex	*[LogIndex]*		

Example URL

.../HistoricalData/v2/Logs/6a684584-58a1-4ab5-a777-417ac0b2bc58--9--/

Exemple playload (JSON)

```
{
   "LogListName": "Main",
   "AutoRefresh": "1",
   "RefreshPeriod": "5",
   "ElementMaxNumber": "20",
   "properties": [
      "VarName",
      "Description"
   ],
   "eventCodes": [
      "BitChangeTo0",
      "BitChangeTo1",
      "AlarmOnNotAck",
      "AlarmOffNotAck",
      "AlarmOnAck",
      "AlarmOffAck"
   ]
}
```

Responses:

Response (JSON)

```
{
   "code": {
      "value": 1,
      "label": "S_Ok"
   },
   "Description": null
}
```

DELETE	/HistoricalData/v2/Logs/{RequestId}	Deleting a log request

Parameters:

Name	Value	Description	
{RequestId}	*string*	Request Id	*mandatory*
Example URL			
.../HistoricalData/v2/Logs/6a684584-58a1-4ab5-a777-417ac0b2bc58--9--/			

Responses:

Response (JSON)

```
{
   "code": {
      "value": 1,
      "label": "S_Ok"
   },
   "Description": null
}
```

4.6 Schemas

SVLanguage

Name	*Value*	*Description*
ContextDefault	0	
BaseLanguage	1	
AlternativeLanguage	2	

LogIndex

Name	*Value*	*Description*
Id	*string*	
LowEventValue	*Integer*	
HighEventValue	*Integer*	
Station	*Integer*	
ItemSearch	*boolean*	

TrendProperty

Name	*Value*	*Description*
VariableName	0	
Description	1	
AssocLabel	2	
StandardLabel	3	
UserName	4	
Station	5	
AlarmLevel	6	
Informations	7	
LogInformations	8	
TimestampType	9	
TextAttr01 → TextAttr16	10 → 25	
DeferredTextAttr03 → DeferredTextAttr16	26 → 39	
BinAttr	40	

LogProperty

Name	*Value*	*Description*
VariableName	0	
Description	1	
AssocLabel	2	
StandardLabel	3	
UserName	4	
Station	5	
AlarmLevel	6	
Informations	7	
LogInformations	8	
TimestampType	9	
TextAttr01 → TextAttr16	10 → 25	
DeferredTextAttr03 → DeferredTextAttr16	26 → 39	
BinAttr	40	

LogSupportedEventCode

Name	*Value*	*Description*
BitChangeTo0	0	
BitChangeTo1	1	
BitUnavailable	2	
AlarmOnNotAck	3	
AlarmOffNotAck	4	
AlarmOnAck	5	
AlarmOffAck	6	
AlarmUnavailable	7	
AlarmOn	8	
AlarmOff	9	
AlarmNotAccessible	10	
AlarmInhibited	11	
AlarmProgramMasked	12	
AlarmVariableMasked	13	
AlarmUserMasked	14	
AlarmExpressionMasked	15	
UserActionSendCommand	16	
UserActionAlarmAcknowledgement	17	
UserActionAlarmMaskUnmask	18	
UserActionLogonLogoff	19	
UserActionSendProgram	20	

ARC Informatique

Headquarters and Paris offices
2 avenue de la Cristallerie
92310 Sèvres - France

tel + 33 1 41 14 36 00
fax + 33 1 46 23 86 02
hotline +33 1 41 14 36 25
arcnews@arcinfo.com
www.pcvuesolutions.com

ARC Informatique
Private limited company
capitalized
at 1 250 000 €
RCS Nanterre B 320 695 356
APE 5829C
SIREN 320 695 356
VAT N°FR 19320695356

SV Web Services Toolkit REST API

ISO 9001 and ISO 14001 certified