

# NetBrain Single Pane of Glass (SPoG) Integration\_PRTG

NetBrain applicable versions: 7.1a1, 8.0

## Contents

Instruction.....	2
What is NetBrain Single Pane of Glass (SPoG)?.....	2
How does NetBrain SPoG work? .....	2
PRTG API.....	2
Accessing Live Object Data and Live Status Data .....	2
Create NetBrain API Parser.....	4
Define NetBrain API Plugin.....	5
Test NetBrain API Server Instance Connectivity to PRTG Instance .....	5
Adding an External API Server.....	5
Test External API server result .....	5
Create NetBrain Qapp with NetBrain API Parser .....	6
General NetBrain Data View by NetBrain Qapp .....	6
Appendix .....	6
NetBrain API Plugin Code Standard .....	6
NetBrain API Parser Code Standard.....	6

## Instruction

### What is NetBrain Single Pane of Glass (SPoG)?

NetBrain integrates with different data sources within an enterprise to use NetBrain map and Qapp for data correlation, analysis, and troubleshooting.

### How does NetBrain SPoG work?

NetBrain has Python function defined in API Plugin to send HTTP/HTTPS request to 3<sup>rd</sup> party system to query 3<sup>rd</sup> party system data via REST API. End user needs to specify the 3<sup>rd</sup> party system REST API for certain corresponding data in NetBrain API Parser. NetBrain will then be able to implement the parser in a Qapp to further process the REST API retrieved data to generate NetBrain Data View on NetBrain maps.

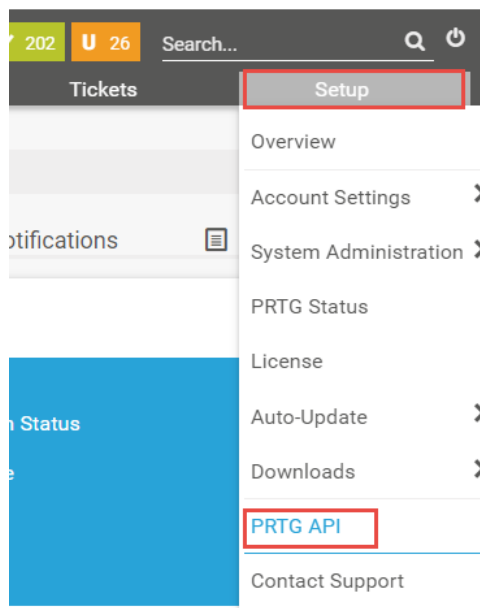
This documentation uses display PRTG device's sensor data as NetBrain Data View (DV) as an example to explain how PRTG SPoG is implemented in NetBrain system.

## PRTG API

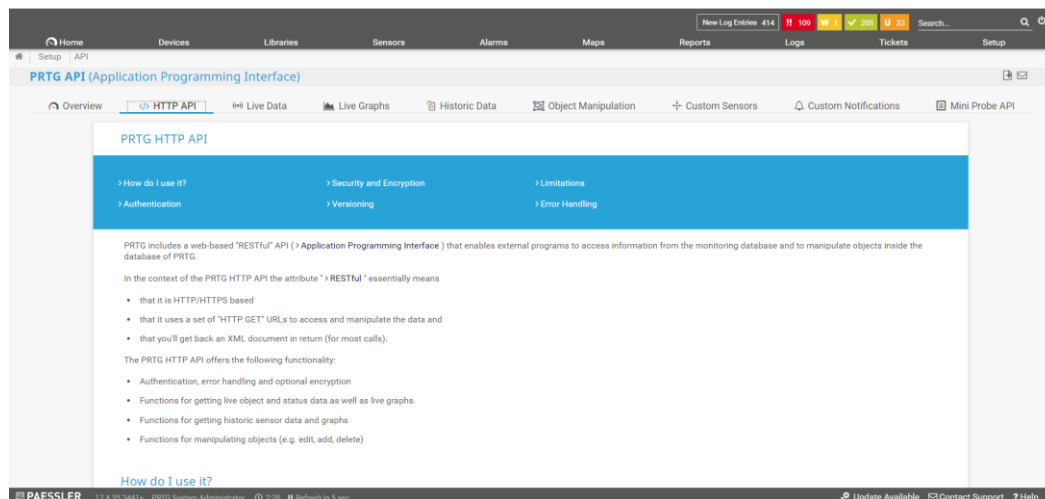
### Accessing Live Object Data and Live Status Data

To define NetBrain API Parser, end user needs to first understand what data needs to be pulled from PRTG and the corresponding PRTG API to query such data.

Navigate to "PRTG API" page in PRTG.



Brief introduction of using PRTG API



## 1. PRTG HTTPAPI

PRTG includes a web-based "RESTful" API ( Application Programming Interface ) that enables external programs to access information from the monitoring database and to manipulate objects inside the database of PRTG. The URLs consist of a path to the API function and some parameters. Here are two example calls:

Sample Call 1:

`http://yourserver/api/table.json?content=sensortree`

Sample Call 2:

`http://yourserver/api/rename.json?id=objectid&value=newname`

## 2. Check if data is in correct format, get path parameters and Query parameters.

Navigate to "PRTG HTTP API: XML Table Query Builder" to check if table query builder shows the JSON data returned from the API server.

## XML Table Query Builder

Please choose from the available contents for tables

Table Content	Table Type
sensortree: A tree-like structure of groups, devices and sensors	xml: most suitable for further processing (recommended)
groups: List of all groups	<b>json: lightweight javascript object notation</b>
devices: List of all devices	xmtable: structured like an HTML table (easier to convert into a table)
<b>sensors: List of all sensors</b>	csv: comma separated output
values: List of most recent results of a sensor (sensor ID required)	html: pure HTML
channels: List of the channels of a sensor (sensor ID required)	
reports: List of reports	
storedreports: List of most recent PDF reports (report ID required)	
tickets: List of most recent tickets	

Columns	Count	Start
objid,probe,group,device,sensor,status,message,lastvalue,priori	Maximum number of items (default 500)	Start with this entry number (can be used with "count" to request the data page-by-page)
Comma delimited list of columns per record. Please see columnslist below for details		
Object ID		
The table will only contain information for this object id and its child objects (all objects will be used if this parameter is omitted)		

### Run query and preview output

Your Query URL

/api/table.json?content=sensors&output=json&columns=objid,probe,group,device,sensor,status,message,lastvalue,priority,favorite

Query Results

```
[{"prtg-version":"17.4.35.3441","tree-size":338,"sensors":[{"objid":1001,"probe":"Local Probe","group":"Local Probe","device":"Probe Device","sensor":"System Health","status":"Up","status_raw":3,"message":"<div class='\"status'><div class='\"moreicon'></div></div>","message_raw":"OK","lastvalue":100,"lastvalue_raw":100.0000,"priority":5,"favorite":0,"caption class='\"objectisnotfavorite icon-gray ui-icon ui-icon-flag'\" id='\"fav-1001'\" onclick='\"Prtg.ObjectTools.FavoriteObject.call(this,1001,aspos.toggleaspos);return false;'\"></span>","favorite_raw":0},{"objid":1002,"probe":"Local Probe","group":"Local Probe","device":"Probe Device","sensor":"Core Health","status":"Warning","status_raw":4,"message":"<div class='\"status'>504 d (Age of Code) is above the warning limit of 90 d in Age of Code. Please consider upgrading to the latest version to improve security and stability</div>","message_raw":"504 d (Age of Code) is above the warning limit of 90 d in Age of Code. Please consider upgrading to the latest version to improve security and stability","lastvalue":100,"lastvalue_raw":100.0000,"priority":5,"favorite":0,"caption class='\"objectisnotfavorite icon-gray ui-icon ui-icon-flag'\" id='\"fav-1002'\" onclick='\"Prtg.ObjectTools.FavoriteObject.call(this,1002,aspos.toggleaspos);return false;'\"></span>","favorite_raw":0},{"objid":1003,"probe":"Local Probe","group":"Local Probe","device":"Probe Device","sensor":"Probe Health","status":"Up","status_raw":3,"message":"<div class='\"status'>Disconnected</div>","message_raw":"Disconnected","lastvalue":100,"lastvalue_raw":100.0000,"priority":5,"favorite":0,"caption class='\"objectisnotfavorite icon-gray ui-icon ui-icon-flag'\" id='\"fav-1003'\" onclick='\"Prtg.ObjectTools.FavoriteObject.call(this,1003,aspos.toggleaspos);return false;'\"></span>","favorite_raw":0}]]
```

Get Path parameters and Query parameters

For more details on PRTG HTTP API, refer to <http://YourServer/api.htm?tabid=3>

## Create NetBrain API Parser

The screenshot shows the NetBrain API Parser interface. The top bar includes a search field, a path field, and a user profile. The main area is divided into two panels: "Define function to retrieve data" (left) and "Define function to parse data" (right). The left panel contains a Python function definition for retrieving data from the PRTG API. The right panel contains a JavaScript function definition for parsing the retrieved data. Below these panels, the "Retrieval results" section shows the raw JSON data, and the "Parse Results" section shows the parsed data in a table format.

**Define function to retrieve data:**

```
def BuildParameters(context, device_name, params):
    rtn_params = {}
    rtn_params['api_params'] = {
        'url': '/api/table.json',
        'context': 'sensors',
        'columns': 'objid,probe,group,device,sensor,status,message,lastvalue,priority,favorite',
        'filter_device': device_name
    }
    return (True, rtn_params)

def RetrieveData(rtn_params):
    sensors = get_data(rtn_params)
    return sensors
```

**Define function to parse data:**

```
1 # Declare variable structure, type and name.
2 ...
3 Begin Declare Variable
4 ...
5 {
6     { "name": "num_sensor", "type": "int",
7       { "name": "sensor_detail", "type": "table", "columns": [
8         { "name": "sensor", "type": "string",
9           { "name": "sensor_id", "type": "int",
10            { "name": "status", "type": "string",
11             { "name": "last_value", "type": "string"
12          }
13       }
14     }
15 }
16 ...
17 End Declare
18 ...
19 import json
```

**Retrieval results:**

```
{
  "device": "US-BOS-R1",
  "favorite": "cspan class='\"objectisnotfavorite icon-gray ui-icon ui-icon-flag'\" id='\"fav-4'\"",
  "group": "Unity lab",
  "lastvalue": "0 msec",
  "lastvalue_raw": 0,
  "message": "<div class='\"status'><div class='\"moreicon'></div></div>","message_raw": "OK",
  "objid": 4668,
  "priority": 3,
  "probe": "Local Probe",
  "sensor": "Ping",
  "status": "Up",
  "status_raw": 3
}
```

**Parse Results:**

Original_Result	\$sensor(string)	\$sensor_id(int)	\$status(string)	\$last_value(string)
\$num_sensor(int)	Ping	4668	Up	0 msec
\$Script(\$sensor_detail)	(001) to 192 MG...	4669	Up	0.10 kbit/s
\$sensor(string)	(002) EthernetQ/...	4670	Up	21 kbit/s
\$sensor_id(int)	(003) EthernetQ/...	4671	Up	12 kbit/s
\$status(string)	System Health C...	4672	Up	0 %
\$last_value(string)	System Health M...	4673	Up	757 MByte

### A. Define retrieve data function

#### a. Customize Python functions:

- i. BuildParameters()
 

Define PRTG HTTP API parameters.

    - api\_url: PRTG API HTTP URI without host server domain name section
    - url\_params: PRTG API HTTP Query parameters
  - ii. RetrieveData()
 

Call API Plugin Python function to retrieve data via REST API
- B. Review retrieved data
- C. Define parse data function
- D. Review parsed data

## Define NetBrain API Plugin

NetBrain API Plugin has a build-in instance for PRTG, which contains HTTP Get function (get\_data()).

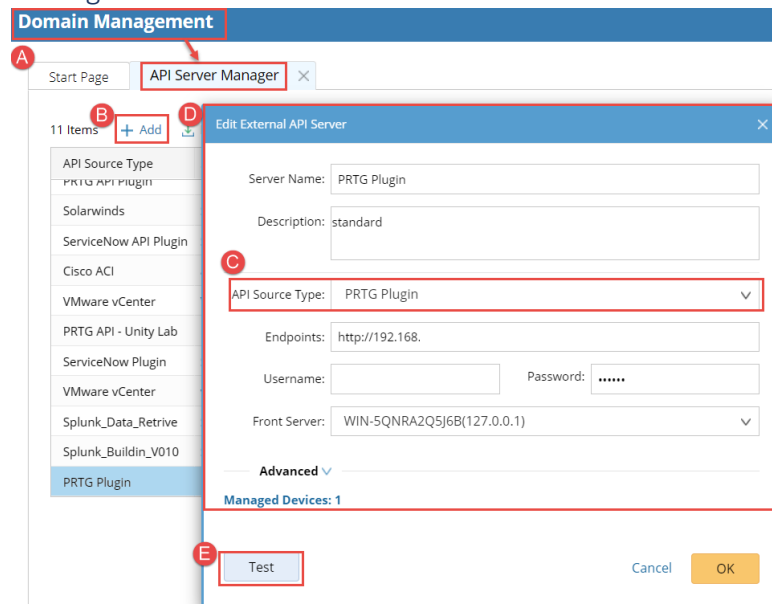
The HTTP request is using Basic Authentication to be authenticated by PRTG for each API call.

The username and password information are inherited from NetBrain **API Server Manager** instance.

No customization and modification are required in API Plugin.

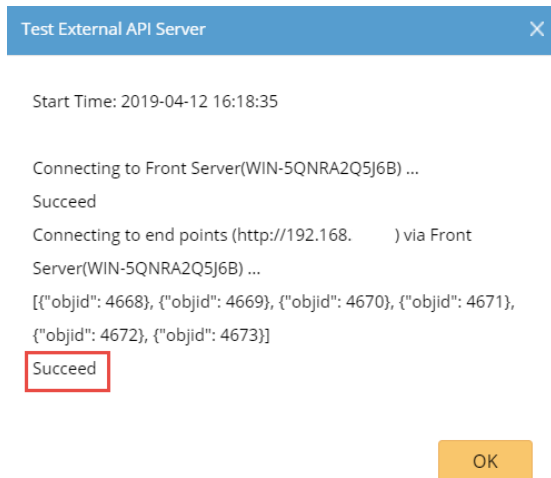
## Test NetBrain API Server Instance Connectivity to PRTG Instance

### Adding an External API Server



- a. Open the Domain Management page, and navigate to **API Server Manager** tab.
- b. Click **Add** button to add an External API Server
- c. Select “**PRTG Plugin**” as **API Source Type**
- d. Fill out all other fields in the pop-up dialog above
- e. Click **Test** button to test connectivity from NetBrain Front Server to PRTG Server IP

### Test External API server result



## Create NetBrain Qapp with NetBrain API Parser

## General NetBrain Data View by NetBrain Qapp

## Appendix

### NetBrain API Plugin Code Standard

1. To easily maintain and scale API Parser Library in the future, only extract parameters passed from Parser. DO NOT hard code any REST API HTTP parameters in API Plugin Python functions.
2. To prevent more HTTP request sending from NetBrain to 3<sup>rd</sup> party systems, use Basic Authentication for HTTP calls as long as 3<sup>rd</sup> party system supports Basic Auth.
3. To easily organize HTTP call parameters, return the following 4 values to get\_data() function:
  - a. endpoint: 3<sup>rd</sup> party system host address, which is defined in API Server Manager
  - b. username: 3<sup>rd</sup> party system login username to be used by HTTP call, which is defined in API Server Manager
  - c. password: 3<sup>rd</sup> party system login password to be used by HTTP call, which is defined in API Server Manager
  - d. api\_params: HTTP request parameters defined in API Parser
4. To simplify the output in API Server Manager Test result, trim the sample REST API call result as much as possible in \_test() function.

### NetBrain API Parser Code Standard

1. Define HTTP request parameters in BuildParameters() function by following the Python dictionary structure below:

```
rtn_params['api_params'] = {
    'api_uri': '/api/table.json',
    'url_params': {
        'content': 'sensors',
        'columns': 'objid,probe,group,device,sensor,status,message,lastvalue,priority,favorite',
        'filter_device': 'device_name'
    }
}
```

}

2. Trim the result by better defining the query parameters, instead of further process the returned result later in NetBrain parser.
3. Implement the main logics by calling API Plugin functions in RetrieveData() function