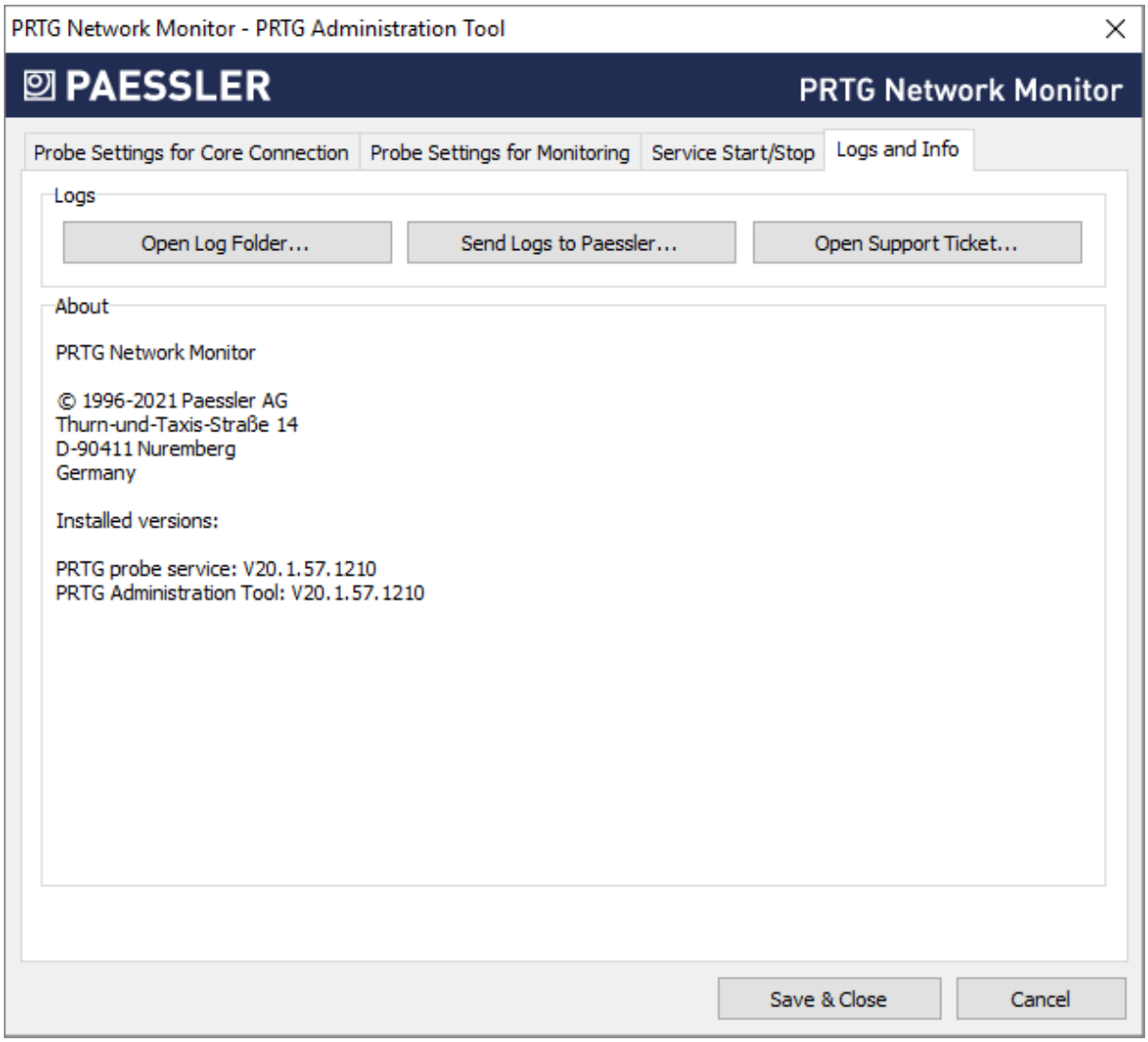
Setting	Description
	▪ Scheduled restart of PRTG services: Restart the PRTG probe service on the probe system. If you select this option on the local probe, the PRTG core server service restarts as well. Define a schedule under Restart Schedule. ▪ Scheduled system restart (recommended): Define a schedule under Restart Schedule. We recommend that you restart probe systems once a month for best performance.
Restart Schedule	This setting is only visible if you select a schedule option above. Choose how often you want to restart the PRTG probe service or the probe system: ▪ Once per week: Select a day and a time below. ▪ Once per month (recommended): Select a day of the month and a time below.
Day	This setting is only visible if you select a schedule option above. Select a day of the week (Monday to Sunday) or month (1st to 30th or Last). If you select Last, PRTG restarts the PRTG core server system on the last day of the month, regardless of how many days the month has. ⓘ If you select a date that does not exist in every month (for example, the 30th day of February), PRTG automatically initiates the restart on the last day of this month.
Time	This setting is only visible if you select Scheduled restart of PRTG services or Scheduled system restart (recommended) above. Select the time of day when PRTG performs the restart. ⓘ You get a Windows warning message 10 minutes before the restart to inform you about the restart if you are logged in to PRTG. The actual restart time can differ by up to 30 minutes from the time you enter here.

ⓘ You can also define a restart schedule on the Settings tab 517 of a remote probe in the PRTG web interface.

Logs and Info



Logs and Info Tab

Logs

Button	Description
Open Log Folder	Open the PRTG data directory 3637 to access all logs that PRTG creates.
Send Logs to Paessler	Open an assistant to send logs to the Paessler support team. See Send Logs to Paessler 3503 for details. ⓘ You can also send logs with the support bundle via Contact Support 3414 in the PRTG web interface.
Open Support Ticket	Open the support form on the Paessler website in a browser window.

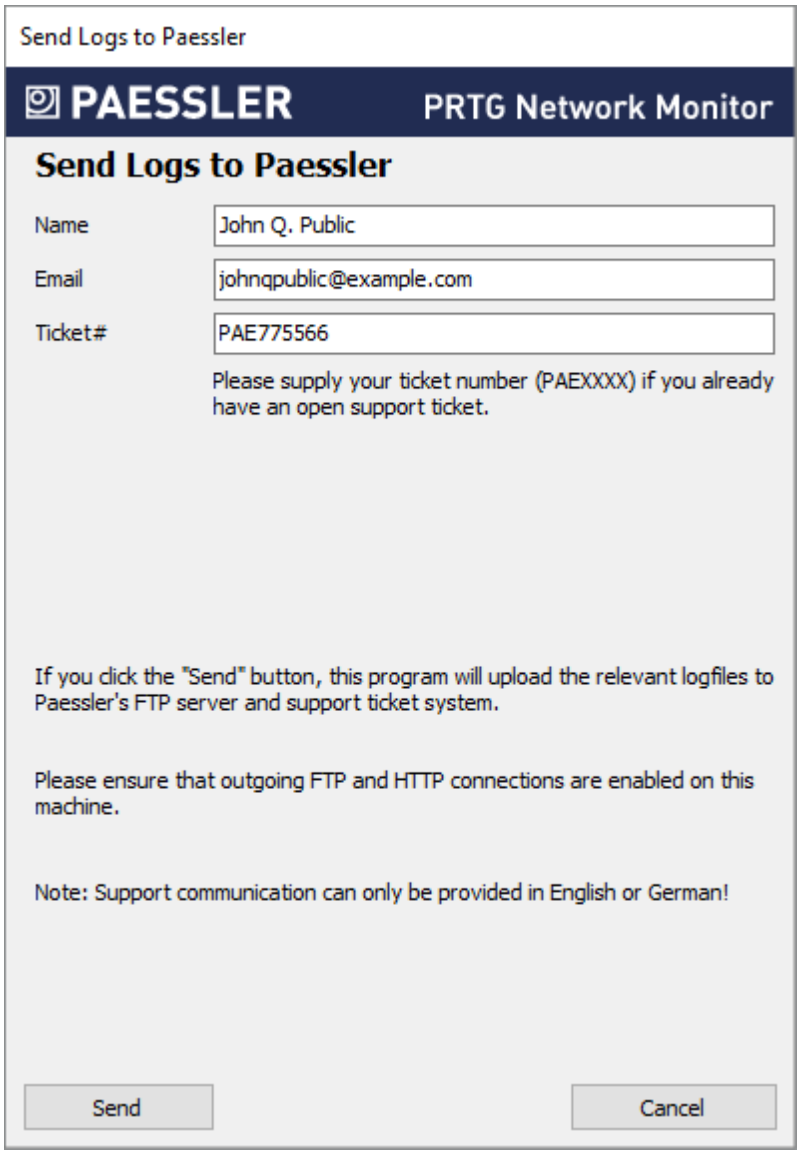
Button	Description
	ⓘ If you need help, we recommend that you use the Contact Support option in the PRTG web interface instead.

About

The About section shows information about the version of installed PRTG programs and copyright information.

Send Logs to Paessler

ⓘ You can also send logs with the support bundle via Contact Support in the PRTG web interface.



Send Logs to Paessler

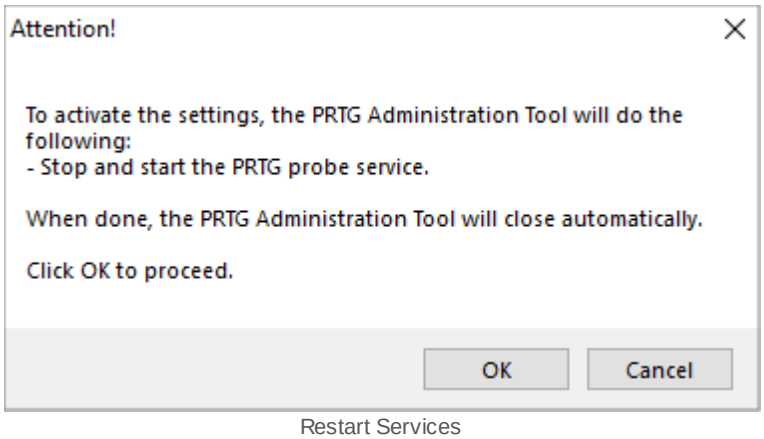
If you open a support ticket, the Paessler support team might ask you to send logs for further analysis.

Field	Description
Name	Enter your name.
Email	Enter a valid email address. You can provide any email address but we recommend that you use the email address of your user account, which PRTG enters by default.
Ticket #	This field is optional. If you have already opened a ticket with the Paessler support team, provide the ticket number you received. Your files are then automatically associated with your ticket. Enter the ticket number starting with PAE followed by four or more digits, for example, PAE12345. If you do not have a ticket number, leave this field empty.

Click Send to start the data upload. PRTG then automatically collects, compresses, and sends your logs to our FTP over SSL (FTPS) server. Ensure that FTPS and HTTP connections are allowed on the remote probe system.

Activate Changed Settings

After you change settings, click Save & Close. A new window opens where PRTG asks you to agree to restart the PRTG probe service. Click OK to proceed.



Restart Services

More

KNOWLEDGE BASE

What security features does PRTG include?

- https://kb.paessler.com/en/topic/61108

Which ports does PRTG use on my system?

- https://kb.paessler.com/en/topic/61462

Part 14
Advanced Topics

14 Advanced Topics

In this section, we cover more advanced topics. If you already have some experience with PRTG, you might want to learn more about the following topics.

In this section:

- Active Directory Integration 3508
- Application Programming Interface (API) Definition 3511
  - HTTP API 3512
  - Live Data 3515
  - Live Graphs 3546
  - Historic Data 3549
  - Object Manipulation 3552
  - Custom Sensors 3559
  - Custom Notifications 3574
  - Mini Probe API 3576
- Filter Rules for Flow, IPFIX, and Packet Sniffer Sensors 3596
- Channel Definitions for Flow, IPFIX, and Packet Sniffer Sensors 3601
- Define IP Address Ranges 3603
- Define Lookups 3604
- Regular Expressions 3616
- Calculating Percentiles 3618
- Add Remote Probe 3619
  - Remote Probes and Multiple Probes 3621
  - Remote Probe Setup via Device Tools 3625
- Failover Cluster Configuration 3631
- Data Storage 3636

14.1 Active Directory Integration

You can add PRTG user groups to PRTG, or you can add user groups from an Active Directory (AD). When you integrate the AD into PRTG, you map a user group from the AD to a user group in PRTG. All members of the AD group can then log in to PRTG with their AD domain credentials.

ⓘ You cannot add single AD users to PRTG. You can only allow access for entire AD groups. PRTG automatically creates a user account for each AD user that successfully logs in to PRTG.

☁ This feature is not available in PRTG Hosted Monitor.

Step 1: Prepare the AD

- In the AD, make sure that the users that you want to give access to PRTG are members of the same user group in the AD.
- You can also organize users into different user groups, for example, one user group whose members have administrative rights in PRTG, and one user group whose members only have read access in PRTG.

Step 2: Prepare the PRTG Core Server

- Make sure that the PRTG core server system is a member of the domain that you want to integrate it into. To check this setting, open the Windows Control Panel and click the Change settings link under System, section Computer name, domain, and workgroup settings.

Step 3: Add AD Domain and Credentials (optional) to System Settings

- In the PRTG web interface 124, select Setup | System Administration | Core & Probes from the main menu.
- In section Active Directory Integration, enter the name of the local AD domain in the Domain Name field.
  ⓘ You can only integrate one AD domain into PRTG.
- The following process is optional. PRTG uses the same Windows user account from which a user runs the PRTG core server service. By default, this is the local system Windows user account. If this user does not have sufficient rights to query a list of all user groups from the AD, provide the credentials of a user account that has full AD access by using the Use explicit credentials option as Access Type.
  ⓘ If you cannot save changes to the Core & Probes settings because you get an Error (Bad Request) with the message Active Directory Domain not accessible, select Use explicit credentials as Access Type and provide the correct credentials for your AD domain.
- Save your settings.

Step 4: Add a New User Group

- Go to the User Groups tab 3346.
- Hover over ⊕ and click Add User Group to add a new user group.
- Enter a User Group Name to identify the group and select Use Active Directory integration under Active Directory or Single Sign-On Integration.

- From the Active Directory Group dropdown list, select the group in the AD whose members have access to PRTG. If you have a very large AD with more than 1,000 entries, you see an input field instead of a dropdown list. In this case, you can only enter the name of the user group in the AD. PRTG automatically adds the domain name prefix.
- For User Type, define the access rights 144 that a user from the selected AD group has when they log in to PRTG for the first time. You can choose between Read/write user and Read-only user. Read-only access is useful to only show data to a large group of users.
- Click Create.

All users in this newly created AD group can now log in to PRTG with their AD domain credentials. Their user accounts have the group access rights 144 of the user group that you just created.

Notes and Restrictions

- AD users can log in 152 to the PRTG web interface with their Windows user name and password. Do not enter any domain information in the Login Name field. When an AD user logs in, PRTG automatically creates a corresponding local account on the PRTG core server. PRTG synchronizes credentials every hour.
- Do not change the Login Name in PRTG for AD users unless the name changes in the AD. If you change the Login Name in PRTG, it has no effect on the name in the AD.
- AD queries are in read-only mode and are compatible with Read-only Domain Controllers (RODC).
- For performance reasons, PRTG caches all requests to AD servers for one hour. If a password changes, if you add a new user group in the AD, or if you change the group membership of an AD user, you must either wait one hour or manually clear the cache by selecting Setup | System Administration | Administrative Tools from the main menu and clicking Go! in the Clear Caches section.
- By default, no access rights for monitoring objects, libraries, maps, or reports are set for the new user group in PRTG. This is why, initially, users in this user group do not see monitoring objects, libraries, maps, or reports. This does not apply if the new user group has administrative rights. Edit the monitoring object's settings 198 and the settings of libraries, maps, and reports, and set access rights for the newly created user group in the respective Access Rights section.
  ⓘ We recommend that you set these access rights in the root group settings 452 and use the inheritance of settings 135.
- PRTG only supports explicit user group rights. If the AD uses groups that are members of other user groups, PRTG does not regard the inherited implicit rights of the parent group and therefore refuses the login for members of these user groups.
- PRTG ignores AD information about organization units (OU). PRTG cannot read these values. However, if you use the AD in an auto-discovery group 272, you can restrict the auto-discovery to machines that are part of an OU.
- You can integrate only one AD domain into PRTG.
- PRTG does not support trusted domains or AD subdomains.
- If you have a very large AD with more than 1,000 entries, you see an input field instead of a dropdown list. In this case, you can only enter the name of the user group in the AD. PRTG automatically adds the domain name prefix.
- A local user account for an AD user is only created if this AD user has successfully logged in to PRTG. If you want to send email notifications 3252 to an AD group in PRTG using the option Send to User Group in the notification settings, a member of this AD group has to log in to PRTG at least once to receive email notifications. To avoid this, enter the email address of the AD group in the Send to Email Address field in the notification settings and select None for the Send to User Group option.

- If you want to delete an AD group from PRTG because of some changes to the AD, for example, you must delete all users that are in this user group first. This is because AD users always have this user group set as their primary group, and user accounts must have a primary group.
- If you change the group membership of an AD user, this change is only reflected in the respective user groups in PRTG if this AD user has logged in to PRTG again.
- If you delete an AD user from all user groups in the AD that are related to PRTG access, this user cannot log in to PRTG anymore. However, you still see the user in the user account list in PRTG.

More

KNOWLEDGE BASE

How to integrate Azure Active Directory into PRTG?

- https://kb.paessler.com/en/topic/88527

14.2 Application Programming Interface (API) Definition

The PRTG API enables you to access monitoring data and manipulate objects using HTTP requests, run your own written sensors and notifications, and implement mini probes.

In this section:

- HTTP API 3512
- Live Data 3515
  - Single Object Property 3516
  - Single Object Status 3519
  - Multiple Object Property or Status 3526
  - System Information 3542
- Live Graphs 3546
- Historic Data 3549
- Object Manipulation 3552
- Custom Sensors 3559
- Custom Notifications 3574
- Mini Probe API 3576

More

KNOWLEDGE BASE

How can I share my self-written PRTG script/program with other PRTG users?

- https://kb.paessler.com/en/topic/63737

Where can I find PRTG mini probes which are ready to use?

- https://kb.paessler.com/en/topic/61215

14.2.1 HTTP API

All calls to the PRTG API are performed by HTTP GET requests. The URLs consist of a path to the API function and some parameters.

- If you are accessing the PRTG API inside your secure LAN, you can use HTTP. In environments that are not secure (for example, when accessing your PRTG core server via the internet), you should use HTTPS requests to make sure that your parameters and passwords are encrypted. This way, all communication between the PRTG core server and your client is Secure Sockets Layer (SSL) encrypted.
- For every API call, the default limit of items is 500. If you want to receive more items per call, add a count=xxx parameter with enough room for all sensors.
- You must include authentication with API key or user name and passhash 3240 (or user name and password) in each request.
  See section Authentication 3513 for more information.
- All data in the GET parameters must be UTF-8-encoded and URL encoded.

Output Formats

Most data that you can request from the PRTG API is available in data tables in the Extensible Markup Language (XML), the JavaScript Object Notation (JSON) format, and the comma-separated values (CSV) format (using the XML format is recommended). Here are some sample calls with different output formats.

Examples

XML

```
https://yourserver/api/table.xml?content=sensors&columns=sensor
```

JSON

```
https://yourserver/api/table.json?content=sensors&columns=sensor
```

CSV

```
https://yourserver/api/table.xml?content=sensors&columns=sensor&output=csvtable
```

The example URLs consist of the following elements.

Element	Description
yourserver	The name of your PRTG server.
/api/table.xml or /api/table.json	Addresses an API function. Here, the function renders a table in the XML format or in the JSON format.
content=sensors	Parameter for additional control. In this case, it includes all sensors in the table.

Element	Description
columns=sensor	Parameter for additional control. In this case, only the names of all sensors are shown in the table.
output=csvtable (optional)	Renders a table in the CSV format.

Authentication

All requests to the PRTG API are stateless, which means that there is no multi-step login process of any kind. You must include the authentication with an API key or with a user name and a passhash (or a user name and a password) in each request by using the apitoken parameter or the user name and passhash (or user name and password) parameters:

apitoken parameter

- apitoken=myapitoken

You can add, edit, and delete your API keys in the account settings on the API Keys 3289 tab.

user name and passhash (or user name and password) parameters

- username=myuser&passhash=hash (or password=mypassword)

You can request the passhash for an account with the following API call:
https://yourserver/api/getpasshash.htm?username=myuser&password=mypassword
Make sure that user name and password are URL-encoded.

Examples

```
https://yourserver/api/table.xml?content=sensors&columns=sensor&apitoken=myapitoken
```

or:

```
https://yourserver/api/table.xml?
content=sensors&columns=sensor&username=myuser&password=mypassword
```

or:

```
https://yourserver/api/table.xml?
content=sensors&columns=sensor&username=myuser&passhash=hash
```

Versioning

Most XML replies from the PRTG API contain a <version> field that contains the program version and buildnumber of the server's PRTG installation. Your client must look at this version number and compare it to the version number that was used to develop the client. Do not accept version numbers older (smaller) than this one. You should display a warning to the user (or stop processing) if the version number differs by 0.1 or more (for example, version 18.1 vs. 18.2, or version 17.x vs. 18.x). PRTG API conventions or parameters might have changed between versions.

ⓘ Newer versions of the same major version of PRTG reply to API calls just like previous versions did.

Error Handling

Depending on whether an API call was successfully processed or not, the PRTG core server replies with the following HTTP status codes:

HTTP Status Code	Meaning	Comments
200	OK	The API call was successfully completed , the XML response contains the result data.
302	Found	The API call was successfully completed and a new object was created (the redirection URL contains the new object ID).
400	Bad Request	The API call could not be successfully completed. The XML response contains the error message.
401	Unauthorized	The user name/password credentials cannot be accepted.

For 400 error codes, the error .xml document includes the error message as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <prtg>
  <version>18.1.37.10127+</version>
  <error>Sorry, there is no object with the specified id.</error>
  </prtg>
```

More

KNOWLEDGE BASE

How can I use the PRTG Application Programming Interface (API)?

- https://kb.paessler.com/en/topic/593

14.2.2 Live Data

You can access live data and live status data of objects using the PRTG API.

In this section:

- Single Object Property 3516
- Single Object Status 3519
- Multiple Object Property or Status 3526
- System Information 3542

Getting PRTG System Status

You can also get the PRTG system status such as the number of alarms or messages using the following API calls.

Examples

Live system status in the Extensible Markup Language (XML) format:

```
/api/getstatus.xml?id=0
```

Live system status in the JavaScript Object Notation (JSON) format:

```
/api/getstatus.htm?id=0
```

All sensors in use in the JSON format:

```
/api/sensortypesinuse.json
```

Number of all sensor states:

```
/api/gettreenodestats.xml
```

Getting PRTG Health Status

You can also get the overall PRTG system health status such as probe connection status and if login is possible using the following API calls:

Examples

Live overall system status of PRTG in the JSON format:

```
/api/healthstatus.json
```

Live overall system status of PRTG Hosted Monitor in the JSON format:

```
/api/healthstatus.json&mode=hosted
```

Getting PRTG Health Data

You can also get the PRTG health data such as system CPU used (%), system memory used (%), disk space used (%), disk space used (GB), health (%), total number of probes, disconnected probes, total sensors, and sensor in the Unknown status using the following API calls:

Example

Live health data of PRTG in the JSON format using maxage:

```
/api/health.json&maxage=age
```

maxage is the age in seconds for data to be considered "old". For example, if data is older than 4 minutes and maxage=120, the data will be refreshed and then sent to the client.

Live health data of PRTG in the JSON format using refreshnow:

```
/api/health.json&refreshnow=1/anything_else
```

If refreshnow=1, the data will be refreshed before the API call is returned. Also, if refreshnow=1 is present in the API call, maxage will not be considered.

More

KNOWLEDGE BASE

How can I use the PRTG Application Programming Interface (API)?

- https://kb.paessler.com/en/topic/593

14.2.2.1 Single Object Property

You can access live data and live status data of single objects using the PRTG API.

Authentication with API key or user name and passhash 3240 (or user name and password) must always be included in each PRTG API request. See section HTTP API 3513 for more information.

In this section:

- Property/Setting of an Object 3516
- Supported Object Types 3517
- Available Channel Parameters 3518
- Property of a Channel 3518

Property/Setting of an Object

You can get the properties or settings of an object (name, hostname, url) using the following API calls.

Because properties might contain HTML content, we recommend that you include &show=nohtmlencode in all getobjectproperty API calls.

Examples

Get an object property/setting:

```
/api/getobjectproperty.htm?id=objectid&name=propertyname&show=nohtmlencode
```

For propertyname, look at the name of the INPUT fields while editing an object. You can discern the propertyname parameter by opening the Settings tab of an object and looking at the HTML source of the INPUT fields. For example, the INPUT field for the tags of an object has the name tags_. Leave away the underscore _ and use tags as a value for the propertyname parameter.

Get a list of all tags for object ID 1003

```
/api/getobjectproperty.htm?id=1003&name=tags&show=nohtmlencode
```

The Extensible Markup Language (XML) result looks like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <prtg>
      <version>18.3.45.1224+</version>
      <result>probehealthsensor</result>
  </prtg>
```

Supported Object Types

getobjectproperty.htm supports the following object types:

- probe
- group
- device
- sensor
- channel
- library
- map
- notification
- report
- schedule
- user account

getobjectproperty.htm does not support the object types ticket and user group.

Available Channel Parameters

Name of Input Field	Setting Name (as displayed in the PRTG web interface)
name	Name
limitmode	Limit disabled or enabled (0 or 1)
limitmaxerror	Upper Error Limit
limitmaxwarning	Upper Warning Limit
limitminwarning	Lower Warning Limit
limitminerror	Lower Error Limit
limiterrormsg	Error Limit Message
limitwarningmsg	Warning Limit Message

Property of a Channel

With this API call, you can get a sensor's channels settings 3121, for example channel limits. In general, this works like getting properties of any other object. To get channel properties via the PRTG API, you need to provide

- the ID of a sensor (parameter id),
- a subtype (channel for channels), and
- a subid (ID of the channel that you want to edit)

Because properties might contain HTML content, we recommend that you include &show=nohtmlencode in all getobjectproperty API calls.

Examples

Get a channel limit

```
/api/getobjectproperty.htm?
id=sensorid&subtype=channel&subid=channelid&name=limitname&show=nohtmlencode
```

For example, the following API call gets the upper warning limit of a channel with the ID 0 of a sensor with the ID 1003

```
/api/getobjectproperty.htm?
id=1003&subtype=channel&subid=0&name=limitmaxwarning&show=nohtmlencode
```

The XML result looks like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <prtg>
      <version>18.3.45.1224+</version>
      <result>25</result>
  </prtg>
```

More

KNOWLEDGE BASE

How can I use the PRTG Application Programming Interface (API)?

- https://kb.paessler.com/en/topic/593

14.2.2.2 Single Object Status

You can access live data and live status data of single objects using the PRTG API.

Authentication with API key or user name and passhash 3240 (or user name and password) must always be included in each PRTG API request. See section HTTP API 3513 for more information.

In this section:

- Status of an Object 3519
- Supported Object Types 3520
- Supported getobjectstatus Output Columns ("Name=" Parameter) 3520
- Sensor Details 3525
- Ticket Status and Message 3526

Status of an Object

You can get the status information (lastvalue, downtime) of an object using the following API calls:

Examples

Get object status:

```
/api/getobjectstatus.htm?id=objectid&name=columnname
```

The Extensible Markup Language (XML) result looks like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <prtg>
      <version>18.4.45.10127+</version>
      <result>True</result>
  </prtg>
```

ⓘ In some cases, the result can contain HTML formatting elements.

Supported Object Types

getobjectstatus.htm supports the following object types:

- probe
- group
- device
- sensor

Supported getobjectstatus Output Columns ("Name=" Parameter)

With getobjectstatus.htm, you can use the following column names for the name parameter.

ⓘ Only one column name is allowed in the API call.

ⓘ If you want to use combinations of column names or more than one object in a single API call, use a table API call instead. For more information, see section Multiple Object Property or Status 3526.

Column Name	What It Displays	Can Be Used for
objid	ID of the object	probes, groups, devices, sensors
type	Object type or sensor type	probes, groups, devices, sensors
name	Name of the object	probes, groups, devices, sensors
tags	List of all tags. This includes tags from the object itself plus tags that are inherited from parent objects.	probes, groups, devices, sensors
active	True/false depending on whether an object is set to paused by a user	probes, groups, devices, sensors
downtime	Cumulated downtime of the sensor (displayed as percentage of uptime+downtime)	sensors

Column Name	What It Displays	Can Be Used for
downtimetime	Cumulated downtime of the sensor (in minutes/hours)	sensors
downtimesince	Elapsed time since the last Up status of the sensor	sensors
uptime	Cumulated uptime of the sensor (displayed as percentage of uptime+downtime)	sensors
uptimetime	Cumulated uptime of the sensor (in minutes/hours)	sensors
uptimesince	Elapsed time since the last Down status of a sensor	sensors
knowntime	Sum of cumulated uptime and downtime of the sensor ⓘ The output contains HTML.	sensors
cumsince	Time stamp when accumulation of uptimes/downtimes began	sensors
sensor	Name of the sensor	sensors
interval	Effective interval setting for the sensor	sensors
lastcheck	Time stamp of the last sensor result ⓘ The output contains HTML.	sensors
lastup	Time stamp of the most recent Up status of the sensor ⓘ The output contains HTML.	sensors
lastdown	Time stamp of the most recent Down status of the sensor ⓘ The output contains HTML.	sensors
device	For sensors: ID of the associated device	devices, sensors

Column Name	What It Displays	Can Be Used for
	For devices: name of the associated device	
group	For sensors: ID of the associated group For devices: name of the associated group	groups, devices, sensors
probe	Name of the associated probe	probes, groups, devices, sensors
grpdev	Name of the associated device and associated group, separated by slash	sensors
notifiesx	Returns a string containing the number of each trigger type defined for this object. If trigger inheritance is active, it displays Inherited.	probes, groups, devices, sensors
intervalx	Either Inherited or the current interval setting of the object	probes, groups, devices, sensors
dependency	Name of an associated dependency or Parent ⓘ The output contains HTML.	probes, groups, devices, sensors
probegroupdevice	Partial object hierarchy with names of associated device, group, and probe separated by angle brackets (>). ⓘ The output contains HTML. ⓘ probegroupdevice is not available in the CSV format. ⓘ For the complete object hierarchy, use probe, group, device instead.	probes, groups, devices, sensors
status	Integer of the status of the object (0=None, 1=Unknown, 2=Scanning, 3=Up, 4=Warning, 5=Down, 6=No Probe, 7=Paused by User, 8=Paused by Dependency, 9=Paused by Schedule, 10=Unusual, 11=Not Licensed, 12=Paused Until, 13=Down Acknowledged, 14=Down Partial)	probes, groups, devices, sensors

Column Name	What It Displays	Can Be Used for
message	Detailed message of the object ⓘ The output contains HTML.	probes, groups, devices, sensors
priority	Priority setting of the object ⓘ The output contains HTML.	probes, groups, devices, sensors
lastvalue	Last sensor result value	sensors
upsens	Number of sensors in the Up status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy. ⓘ If the count is <1 PRTG returns "".	probes, groups, devices, sensors
downsens	Number of sensors in the Down status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy. ⓘ If the count is <1 PRTG returns "".	probes, groups, devices, sensors
downacksens	Number of sensors in the Down (Acknowledged) status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy. ⓘ If the count is <1 PRTG returns "".	probes, groups, devices, sensors
partialdownsens	Number of sensors in the Down (Partial) status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy. ⓘ If the count is <1 PRTG returns "".	probes, groups, devices, sensors
warnsens	Number of sensors in the Warning status	probes, groups, devices, sensors

Column Name	What It Displays	Can Be Used for
	ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy. ⓘ If the count is <1 PRTG returns "".	
pausedsens	Number of sensors in the Paused status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy. ⓘ If the count is <1 PRTG returns "".	probes, groups, devices, sensors
unusualsens	Number of sensors in the Unusual status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy. ⓘ If the count is <1 PRTG returns "".	probes, groups, devices, sensors
undefinedsens	Number of sensors in an undefined status, like None, Unknown, No Probe ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy. ⓘ If the count is <1 PRTG returns "".	probes, groups, devices, sensors
totalsens	Number of all sensors ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy. ⓘ If the count is <1 PRTG returns "".	probes, groups, devices, sensors
favorite	An exclamation mark (!) if the object is marked as favorite ⓘ The output contains HTML.	groups, devices, sensors
schedule	Name of the associated schedule	probes, groups, devices, sensors

Column Name	What It Displays	Can Be Used for
minigraph	Numeric data for the minigraphs. Numbers are 5-minute averages for the last 24 hours (must be scaled to the maximum of the series). There are two datasets: "|" separates measured value series and error series. ⓘ The output contains HTML.	sensors
comments	Object comments	probes, groups, devices, sensors
host	Hostname or IP address	devices
condition	For probes: If the probe is connected or disconnected (0=Disconnected, 1=Unauthorized, 2=Connected, 3=Banned, 4=Init) For groups: The auto-discovery status	probes, groups
basetype	Object type (string)	probes, groups, devices, sensors
baselink	URL of the object	probes, groups, devices, sensors
icon	URL of the device icon	devices
parentid	ID of the parent object	probes, groups, devices, sensors
location	Location property (used in Geo Maps) ⓘ The output contains HTML.	devices
groupnum devicenum	Number of groups or devices in the probe or group	probes, groups

Sensor Details

You can get details about a sensor (sensortype, interval, uptime) using the following API calls:

ⓘ You need the sensor ID to get details about a sensor. You can find the ID on the sensor's Overview tab or by hovering over a sensor in the device tree, for example.For more information about the Overview tab, see the Knowledge Base: What options do I have to review my monitoring data in detail?

Examples

Get details about a sensor in the XML format:

```
/api/getsensordetails.xml?id=sensorid
```

Get details about a sensor in the JavaScript Object Notation (JSON) format:

```
/api/getsensordetails.json?id=sensorid
```

Ticket Status and Message

The following API calls return status and message of a ticket.

Examples

The status of a ticket:

```
/api/getticketstatus.htm?id=ticketid
```

The subject and assignee of a ticket:

```
/api/getticketmessage.htm?id=ticketid
```

More

KNOWLEDGE BASE

How can I use the PRTG Application Programming Interface (API)?

- https://kb.paessler.com/en/topic/593

What options do I have to review my monitoring data in detail?

- https://kb.paessler.com/en/topic/90007

14.2.2.3 Multiple Object Property or Status

You can access live data and live status data of multiple objects using the PRTG API.

Authentication with API key or user name and passhash 3240 (or user name and password) must always be included in each PRTG API request. See section HTTP API 3513 for more information.

In this section:

- Property or Status of Multiple Objects 3527
- Table Query Builder 3527
- Output Data Format 3527
- RAW Date/Time Format 3528
- Common Data Table Parameters 3528
- Filtering by Object ID 3529

- Sorting and Advanced Filtering 3530
- Supported Output Columns ("columns=" Parameter) 3533

Property or Status of Multiple Objects

Most data that you can request from the PRTG API is available in data tables in the Extensible Markup Language (XML) format, the JavaScript Object Notation (JSON) format, and the comma-separated values (CSV) format (using the XML format is recommended). The API function /api/table.xml is used to access data in tables. Here are some sample calls (URLs are shown without authentication parameters to enhance readability).

ⓘ The example URLs only show the XML URLs. Use the API function /api/table.csv or the output=constable parameter to select the CSV format, or /api/table.json to return the JSON format.

Examples

A hierarchical list of all groups, devices, and sensors with their status information:

```
/api/table.xml?content=sensortree
```

ⓘ sensortree does not support JSON output.

All sensors (with status information):

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

All recent log entries:

```
/api/table.xml?
content=messages&columns=objid,datetime,parent,type,name,status,message
```

You have the easiest start if you use the table query builder 3527 or click , which most data tables have in the PRTG web interface. Navigate to the information that you want to use, click , and you are taken to a URL that renders the content of the table in XML format. You can now use the URL as it is or change various parameters to suit your needs.

For more information on possible raw message status values returned by table-based API calls, see the Knowledge Base: Is there a list of log status values for the PRTG API?

Table Query Builder

You can use the query builder tool to experiment with the PRTG API and to fine-tune your queries. You can find it in the PRTG web interface under Setup | PRTG API, section Live Data.

Output Data Format

XML data from the PRTG API contains the fields that you requested in the columns parameter. In most cases, numeric values are included twice: One field contains the value in human-readable format and an additional _RAW field contains the value as a number, which is better suited for further processing and calculations.

Example

```
<status>Up</status>
<status_RAW>3</status_RAW>
<lastvalue>98 %</lastvalue>
<lastvalue_RAW>97.7583</lastvalue_RAW>
<message>Created.<br/>18.3.43.1360</message>
<message_RAW>Created. 18.3.43.1360</message_RAW>
```

- The status field shows the value Up (the according RAW value is 3).
- The lastvalue field shows the value 98% (the according RAW value is 97.7583).
- The message field shows the text Created.
18.3.43.1360 (the according RAW value is Created. 18.3.43.1360).

Strings returned from _RAW columns are surrounded by double quotation marks (") in JSON output.

RAW Date/Time Format

For columns with date/time value, the RAW value is defined as follows: The integral part of a value is the number of days that have passed since Dec 30th, 1899. The fractional part of a value is a fraction of a 24-hour day that has elapsed. To find the fractional number of days between two dates, subtract the two values. Similarly, to increment a date and time value by a certain fractional number of days, add the fractional number to the date and time value.

Here are some examples of date/time RAW values and their corresponding dates and times:

RAW Date/Time Value	Description
0	12/30/1899 00h00m (12:00 midnight)
2.75	1/1/1900 18h00m (6:00 pm)
35065	1/1/1996 00h00m (12:00 midnight)

Common Data Table Parameters

The following parameters are common to all data table API calls:

Parameter	Description	Possible Values
content	Select the objects that you want to have in your table.	sensortree (JSON output not supported)

Parameter	Description	Possible Values
		devices sensors tickets ticketdata messages values channels reports storedreports toplists sysinfo (only JSON output supported)
columns	Comma-separated list of columns per record	see Supported Output Columns ("columns=" Parameter) 3533
output	Control the output format	xml: default format (recommended) xmltable: an HTML table in the XML format csvtable: CSV format html: HTML table json: JSON format jsontable: a table in the JSON format
count	Maximum number of items (The default count is 500)	1-50000
start	Start with this entry number (can be used with "count" to request the data page by page)	any

Filtering by Object ID

Add an ID parameter (for example, id=1) to the API URL to select a subset of items for the data table, for example, to reduce the amount of data transferred for each data table API call. The data table only contains information for this object ID and its child objects.

Some table types require an ID. If you omit the ID parameter or if it has the value zero (0), all available objects are used.

Table Type (content=)	ID Required or Optional	Description	Object Types Allowed for the ID
sensortree	optional	You only get a part of the tree (the object with the specified ID and all child objects below it). ⓘ JSON output is not supported.	Probe Group
sensors devices	optional	You only get the object with the specified ID and all child objects below it.	Probe Group Device
tickets messages	optional	You only get tickets or log file entries that are related to the object with the specified ID or any child objects below it.	Any object
values channels	required	You get the data values (or channels) of the sensor with the specified ID.	Sensor
reports	not used	This data table always includes all reports.	n/a
storedreports	required	You get a list of stored .pdf files of the report selected by the ID.	Report
ticketdata	required	You get the history of the ticket selected by the ID.	Ticket
sysinfo	required	You get system information of the object with the specified ID.	Device

Sorting and Advanced Filtering

There are various options to further filter and to sort the data for each data table API call:

Parameter	Description	Possible Values
filter_drel	Only include records younger than this setting	today yesterday

Parameter	Description	Possible Values
	For content=messages and content=tickets only.	7days 30days 12months 6months
filter_status	Only include sensors with a specific status. Using multiple filter_status fields performs a logical OR. For content=sensors only.	Unknown=1 Collecting=2 Up=3 Warning=4 Down=5 NoProbe=6 PausedbyUser=7 PausedbyDependency=8 PausedbySchedule=9 Unusual=10 PausedbyLicense=11 PausedUntil=12 DownAcknowledged=13 DownPartial=14
filter_tags	Only include sensors with a specific tag. Using multiple filter_tag fields performs a logical OR. For content=sensors only.	@tag(tagname)
filter_xyz	Filter the data. (Samples: filter_type=ping, filter_favorite=1). Using multiple filter_xyz fields performs a logical OR.	filter_xyz where xyz is any column name used in the columns parameter Substrings: use filter_xyz=@sub(substring1,substring2) Values not equal/above/below: use filter_xyz=@neq(value), filter_xyz=@above(value), filter_xyz=@below(value)

Parameter	Description	Possible Values
	Filtering using columns is only possible for tree objects. You cannot use columns to filter objects like messages or tickets, for example. For content=tickets, you can use the special filter terms filter_status filter_user, and filter_type (this corresponds to column tickettype). Like for messages, you can also use filter_drel, filter_dstart, and filter_dend. ⓘ Multiple filters are not available for tickets.	
sortby	Sort the data. If this parameter is omitted, the table is sorted based on the first column. Add a leading "-" to reverse sort order. (Samples: sortby=name, sortby=lastvalue, sortby=-lastvalue, sortby=uptime)	Any column name used in the columns parameter. ⓘ Log tables with content=messages are always sorted by descending date.

Examples

Here are some samples for filtered API calls:

All sensors that are not in the Up status 179 (with their status and downtime information):

```
/api/table.xml?
content=sensors&columns=objid,downtimesince,device,sensor,lastvalue,status,message,
priority
&filter_status=5&filter_status=4&filter_status=10&filter_status=13&filter_status=14
&sortby=priority
```

Fastest Ping sensors:

```
/api/table.xml?
content=sensors&columns=objid,sensor,lastvalue,status,message&sortby=lastvalue
&filter_type=ping
```

Log entries of the last 7 days for object id 2003:

```
/api/table.xml?content=messages&id=2003&start=0&filter_drel=7days&columns=
objid,datetime,type,name,status,message
```

Supported Output Columns ("columns=" Parameter)

You can use the following column names for the columns parameter (separated by comma, for example, columns=objid,name,type).

Column Name	What It Displays	Can Be Used for
objid	ID of the object	all object tables
type	Object type (group, device, report, etc.), or the sensor type (ping, http, etc.), or event type for tickets (relevant for ToDo tickets)	all object tables
name	Name of the object or channel. For log messages/tickets: the name of the related object. For stored reports: the name of the report file.	all object tables channels messages storedreports toplists tickets
tags	List of all tags (for tickets: tags for the related object). This includes tags from the object itself plus tags that are inherited from parent objects.	all object tables (except for user)
active	True/false depending on whether an object is set to paused by a user (for tickets: related object). For notifications that are paused by schedule, it also displays the end of the schedule.	all object tables
downtime	Cumulated downtime of the sensor (displayed as percentage of uptime+downtime)	sensors
downtimetime	Cumulated downtime of the sensor (in minutes/hours)	sensors

Column Name	What It Displays	Can Be Used for
downtimesince	Elapsed time since last Up status of the sensor	sensors
uptime	Cumulated uptime of the sensor (displayed as percentage of uptime+downtime)	sensors
uptimetime	Cumulated uptime of the sensor (in minutes/hours)	sensors
uptimesince	Elapsed time since the last Down status of the sensor	sensors
knowntime	Sum of cumulated uptime and downtime of the sensor	sensors
cumsince	Time stamp when accumulation of uptimes/downtimes began	sensors
sensor	Name of the sensor	sensors toplists
interval	Effective interval setting for the sensor	sensors
lastcheck	Time stamp of the last sensor result	sensors
lastup	Time stamp of the most recent Up status of the sensor	sensors
lastdown	Time stamp of the most recent Down status of the sensor	sensors
device	Name of the associated device	sensors devices
group	Name of the associated group	sensors devices groups
probe	Name of the associated probe	sensors

Column Name	What It Displays	Can Be Used for
		devices groups probes
grpdev	Name of the associated device and the associated group separated by a forward slash (/)	sensors devices
notifiesx	Number of each trigger type defined for the object	probes groups devices sensors
intervalx	Either Inherited or the current interval setting of the object	probes groups devices sensors
accessrights	Access rights of the current user for the sensor tree object	all objects (except for user), for example: probes groups devices sensors
dependency	Name of an associated dependency or Parent	probes groups devices sensors
probegroupdevice	Partial object hierarchy with names of associated device, group, and probe separated by a forward slash (/). ⓘ The output contains HTML. ⓘ probegroupdevice is not available in the CSV format.	sensor device group probe

Column Name	What It Displays	Can Be Used for
	ⓘ For the complete object hierarchy, use probe, group, device instead.	
status	For sensor tree objects: status of the object (0=None, 1=Unknown, 2=Scanning, 3=Up, 4=Warning, 5=Down, 6=No Probe, 7=Paused by User, 8=Paused by Dependency, 9=Paused by Schedule, 10=Unusual, 11=Not Licensed, 12=Paused Until, 13=Down Acknowledged, 14=Down Partial) For messages: category of the log message For tickets: status of ticket (open, resolved, closed)	sensors devices groups probes messages tickets
message	Detailed message of the sensor tree object (for example, last error of the sensor) or the history entry, log entry, ticket subject	sensors devices groups probes messages tickets ticketdata history
priority	Priority setting of the sensor tree object or the priority of the log entry/ticket	sensors devices groups probes messages tickets (not supported: schedule, notification, user)
lastvalue	Last sensor result value or channel values	sensors channels

Column Name	What It Displays	Can Be Used for
	When used with channels, you must use lastvalue_ to automatically display volumes and speed.	
upsens	Number of sensors in the Up status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy.	all sensors devices groups probes
downsens	Number of sensors in the Down status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy.	all sensors devices groups probes
downacksens	Number of sensors in the Down (Acknowledged) status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy.	all sensors devices groups probes
partialdownsens	Number of sensors in the Down (Partial) status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy.	all sensors devices groups probes
warnsens	Number of sensors in the Warning status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy.	all sensors devices groups probes
pausedsens	Number of sensors in the Paused status. This includes all Paused states ('paused by user', 'paused by dependency, 'paused by schedule', etc.).	all sensors devices groups

Column Name	What It Displays	Can Be Used for
		probes
unusualsens	Number of sensors in the Unusual status ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy.	all sensors devices groups probes
undefinedsens	Number of sensors in an undefined status, like None, Unknown, No Probe ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy.	all sensors devices groups probes
totalsens	Number of all sensors ⓘ PRTG only counts the sensor itself or sensors below the object in the hierarchy.	all sensors devices groups probes
size	Performance impact of the sensor (1=Very Low, 2=Low, 3=Medium, 4=High, 5=Very High)	sensors
value	The channel value or the Toplist value Should only be used as value_, because then it is expanded for all visible channels/toplist columns	values topdata
coverage	Sensor coverage of the time span in a value table.	values
favorite	An exclamation mark (!) if the sensor tree object is marked as favorite	sensors devices groups probes

Column Name	What It Displays	Can Be Used for
user	User responsible for a history entry or the user (or user group) a ticket is assigned to.	history tickets ticketdata
parent	Name of the parent object of the associated object of a log message	messages
datetime	Time stamp or time span of the object (for tickets: last modification)	messages tickets ticketdata values history storedreports topidx
dateonly	Like datetime but only the date part	messages tickets history values
timeonly	Like datetime but only the time part	messages tickets history values
schedule	For sensor tree objects: Name of the associated schedule For reports: Report generation schedule	probes groups devices sensors reports
period	Period of the report (day, week, etc.)	reports
email	Email address of the report	reports

Column Name	What It Displays	Can Be Used for
template	Template used by the report	reports
lastrun	Time stamp of the last generation of a report	reports
nextrun	Time stamp of the next generation of a report	reports
size	Size of a stored report	size of a stored report
minigraph	Numeric data for the minigraphs. Numbers are 5-minute averages for the last 24 hours (must be scaled to the maximum of the series). There are two datasets: "|" separates measured value series and error series.	sensors
deviceicon	Device icon	devices
comments	Object comments For tickets: related object	all objects
host	Hostname or IP address	devices
devices	For probes: probe status (0=Disconnected, 1=Unauthorized, 2=Connected, 3=Banned, 4=Init) For groups: auto-discovery status	probes groups
basetype	Object type (string)	all tree objects
baselink	URL of the object	all tree objects
icon	URL of the device icon	devices
parentid	ID of the parent object or ID of a ticket	all tree objects tickets

Column Name	What It Displays	Can Be Used for
location	Location property (used in Geo Maps)	devices
fold	Subobjects are folded up (true) or down (false) For tickets: user (or user group) to which a ticket is assigned read it since last change	probes groups tickets
groupnum devicenum	Number of groups or devices in the probe or group	probes groups
tickettype	Type of ticket: user, notification, todo	tickets
modifiedby	User who edited the ticket most recently	tickets ticketdata
actions	Types of all ticket edits	ticketdata
content	Text of the ticket that was added with the last edit	ticketdata
channel	Number of channels with an ID greater than or equal to 0	sensors
_key, _value	Key value pair from the system table	sysinfo (category: system)
_displayname, _class, _caption	Display name, class, and caption from the system table	sysinfo (category: hardware)
_user, _domain	User and domain pair from the system table	sysinfo (category: loggedonusers)
_displayname, _creationdate, _processid	Display name, creation date, and process id from the system table	sysinfo (category: processes)
_displayname, _state, _startmode	Display name, state, and start mode from the system table	sysinfo (category: services)

Column Name	What It Displays	Can Be Used for
_displayname, _version	Display name and version pair from the system table	sysinfo (category: software)

More

KNOWLEDGE BASE

Is there a list of log status values for the PRTG API?

- https://kb.paessler.com/en/topic/76501

How can I use the PRTG Application Programming Interface (API)?

- https://kb.paessler.com/en/topic/593

14.2.2.4 System Information

You can access live data and live status data for system information using the PRTG API.

Authentication with API key or user name and passhash 3240 (or user name and password) must always be included in each PRTG API request. See section HTTP API 3513 for more information.

To retrieve system information using API calls, we recommend that you use the following order:

1. Refresh the system information so that it is up to date, if necessary.
   For details, see Scan Now 3542.
2. Retrieve generic system information to ensure that the last scan (step 1) was successful and that there are no errors.
   For details, see Generic Data 3543.
3. Retrieve system information in the form of data tables.
   For details, see Data Tables 3544.

System information API calls only support the JavaScript Object Notation (JSON) output.

Scan Now

The following API calls to retrieve new information for a system information 202 category. To refresh system information via the API, you need to provide

- the ID of a device (parameter id), and
- a kind (system information category 3543).

We recommend that you only refresh system information if absolutely necessary because the refresh triggers a rescan of all system information tables.

Examples

Refresh process information

```
/api/sysinfochecknow.json?id=deviceid&kind=processes
```

Refresh hardware information

```
/api/sysinfochecknow.json?id=deviceid&kind=hardware
```

sysinfochecknow only supports JSON output.

Supported Output Columns ("kind=" Parameter)

Column Name	Category (as displayed in the PRTG web interface)
system	System
hardware	Hardware
processes	Processes
services	Services
software	Software
loggedonusers	Users

Generic Data

The following API calls retrieve generic data about the system information category since the last scan, for example time stamps and if the last scan was successful. To retrieve this information via the PRTG API, you need to provide

- the ID of a device (parameter id), and
- a kind (system information category).

Examples

Users

```
/api/sysinfo.json?id=deviceid&kind=loggedonusers
```

Services

```
/api/sysinfo.json?id=deviceid&kind=services
```

sysinfo only supports JSON output.

Data Tables

The following API calls retrieve all information from a system information category table. To retrieve this information via the PRTG API, you need to provide

- the content type (always sysinfo),
- the category (as displayed in the PRTG web interface),
- columns (see Supported Output Columns ("columns=" Parameter) 3544), and
- the ID of a device (parameter id).

Examples

System

```
/api/table.json?
id=deviceid&content=sysinfo&category=system&usecaption=true&headers=key,value&colum
ns=_key,_value
```

Software

```
/api/table.json?
id=deviceid&content=sysinfo&category=software&usecaption=true&headers=key,value&col
umns=_displayname,_version
```

Data tables for system information only support JSON output.

Supported Output Columns ("columns=" Parameter)

You can use the following sysinfo-specific column names for the columns parameter (separated by comma, for example, columns=_key,_value).

For a list of all supported column names, see section Multiple Object Property or Status 3533.

Column Name	Description	Can Be Used for
_key, _value	Key value pair from the system table	sysinfo (category: system)
_displayname, _class, _caption	Display name, class, and caption from the system table	sysinfo (category: hardware)
_user, _domain	User and domain pair from the system table	sysinfo (category: loggedonusers)

Column Name	Description	Can Be Used for
_displayname, _creationdate, _processid	Display name, creation date, and process id from the system table	sysinfo (category: processes)
_displayname, _state, _startmode	Display name, state, and start mode from the system table	sysinfo (category: services)
_displayname, _version	Display name and version pair from the system table	sysinfo (category: software)

More

KNOWLEDGE BASE

How can I use the PRTG Application Programming Interface (API)?

- https://kb.paessler.com/en/topic/593

14.2.3 Live Graphs

You can use live sensor graphs from PRTG in other web pages using the PRTG API. PRTG renders graphs as .png or .svg files. You can include them in other web pages.

Authentication with API key or user name and passhash 3240 (or user name and password) must always be included in each PRTG API request. See section HTTP API 3513 for more information.

Examples

Live graph as a .png file:

```
/chart.png?type=graph&width=300&height=160&graphid=2&id=0
```

Live graph as an .svg file:

```
/chart.svg?type=graph&width=300&height=160&graphid=2&id=0
```

To switch between PNG and SVG images, change the file extension of /chart to .png or .svg.

The URL does not start with /api. When placing these URLs on web pages, keep in mind that the URLs contain the account user name and password/passhash. This can imply security issues. We recommend that you set up a dedicated read-only 144 user account in PRTG that is member of a dedicated user group, for example, that only has read access to the root group and all underlying entries or, even better, only for the object IDs that are used for graph URLs.

Parameters for Live Graph URLs (chart.png or chart.svg)

Parameter	Description
type	Must be graph
graphid	Select time span of the graph: ▪ 0=live ▪ 1=last 48 hours ▪ 2=30 days ▪ 3=365 days
width	Width of the image in pixels
height	Height of the image in pixels
id	The object ID of the desired graph object. This is usually the ID of a sensor.
clgid	The node ID of the cluster node of the desired graph object

Parameter	Description
	ⓘ Use the format clgid=%7BXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX%7D. You can also use clgid=%7B00000000-0000-0000-0000-000000000000%7D to return a graph that has data from all cluster nodes.
graphstyling	Allow control of some graph styles: ▪ Display legend: graphstyling=showLegend%3D%271%27 ▪ Hide legend: graphstyling=showLegend%3D%270%27 ▪ Control font size: graphstyling=baseFontSize%3D%27XX%27 (XX is the font size) ▪ Control legend and font size at the same time: graphstyling=showLegend%3D%271%27+baseFontSize%3D%275%27
bgcolor	Background color of the PNG image, for example, #ffffff. This affects the area that surrounds the graph. ⓘ The value must be URL encoded, for example, %23ffffff.
plotcolor	Color of the graph's plot area, for example, #ffffff. This affects the whole area within the graph box. ⓘ The value must be URL encoded, for example, %23ffffff.
plotcolor1	Alternating color of the graph's plot area, for example, #ffffff. This affects the tiles within the graph box alternating with plotcolor2. The result is a striped graph box. ⓘ The value must be URL encoded, for example, %23ffffff. ⓘ This parameter is overwritten when using the parameter plotcolor.
plotcolor2	Alternating color of the graph's plot area, for example, #ffffff. This affects the tiles within the graph box alternating with plotcolor1. The result is a striped graph box. ⓘ The value must be URL encoded, for example, %23ffffff. ⓘ This parameter is overwritten when using the parameter plotcolor.
gridcolor	Color of grid lines in the graph's plot area, for example, #ffffff. This affects the horizontal and vertical lines within the graph box. ⓘ The value must be URL encoded, for example, %23ffffff.

Parameter	Description
hide	Do not show defined channels in the graph. Use the ID of a channel to hide it. For example, use hide=-4 to not show the Downtime channel in the graph. To hide more than one channel, use commas to separate the IDs.

Retrieving Chart Legends (JSON)

You can show the legend of a sensor graph (channel IDs, colors, units, channel names) in the JavaScript Object Notation (JSON) format.

Example

Chart legend in the JSON format:

```
/api/chartlegend.json?id=sensorid
```

More

KNOWLEDGE BASE

How can I use the PRTG Application Programming Interface (API)?

- https://kb.paessler.com/en/topic/593

14.2.4 Historic Data

You can download the historic monitoring data for one sensor in the Extensible Markup Language (XML) format or the comma-separated values (CSV) format using the following API calls. You can either request the results of each single monitoring request (called raw data) or you can let PRTG calculate averages of the data (for example, hourly or daily averages).

To avoid potential server overload, the number of requestable values per API call is limited by means of automatic averaging as follows:

Minimum Level of Detail (Average Interval)	Maximum Timeframe per API Call
Raw data (all single monitoring requests)	For up to 40 days per API request
60 minutes/1 hour averages	40 to 500 days per API request ⓘ If you try to use a larger time span than 500 days, PRTG automatically reduces it to 365 days.

ⓘ API calls for historic data are limited to 5 requests per minute.

API Calls for Historic Data

ⓘ Authentication with API key or user name and passhash 3240 (or user name and password) must always be included in each PRTG API request. See section HTTP API 3513 for more information.

The API calls for historic data tables look like this:

Examples

XML:

```
/api/historicdata.xml?id=objectid&avg=0&sdate=2018-01-20-00-00-00&edate=2018-01-21-00-00-00
```

CSV:

```
/api/historicdata.csv?id=objectid&avg=0&sdate=2018-01-20-00-00-00&edate=2018-01-21-00-00-00
```

JavaScript Object Notation (JSON):

```
/api/historicdata.json?id=objectid&avg=0&sdate=2018-01-20-00-00-00&edate=2018-01-21-00-00-00&usecaption=1
```

ⓘ You must supply the object ID of a sensor as well as a start date/time sdate and end date/time edate.

If you use the JSON call, additionally provide the parameter usecaption=1 to get more information than just the raw data table.

API Call for Historic Graphs

Historic graphs are also available (in the PNG format):

Example

PNG:

```
/chart.png?id=objectid&avg=15&sdate=2018-01-20-00-00-00&edate=2018-01-21-00-00-
00&width=850&height=270&graphstyling=baseFontSize='12'%20showLegend='1'&graphid=-1
```

Common Parameters for Historic Data API Calls

You can use the following parameters for the graphs and the data tables:

Parameter	Description	Possible Values
id	ID of the specific sensor	integer
sdate	Start of the time span (date and time)	yyyy-mm-dd-hh-mm-ss
edate	End of the time span (date and time)	yyyy-mm-dd-hh-mm-ss
avg	Average interval in seconds; use 0 to download raw data (= results of all single monitoring requests)	integer
width/height	Width and height of the graph in pixels	integer
graphstyling	baseFontSize='x' sets the size of the font showLegend='x' enables (1) or disables (0) the graph legend	baseFontSize='x'% 20showLegend='x'

Historic Data Query Builder

You can also use the historic data reports 183 to manually generate and analyze historic sensor data via the PRTG web interface.

More

KNOWLEDGE BASE

How can I export historic data from the PRTG API?

- https://kb.paessler.com/en/topic/76768

How can I export raw sensor data automatically from PRTG?

- https://kb.paessler.com/en/topic/343

14.2.5 Object Manipulation

You can use the following functions to manipulate objects (URLs are shown without user name or passhash to enhance readability).

In this section:

- Changing Object Settings 3552
- Supported Object Types for rename.htm 3553
- Switch Inheritance Off/On 3553
- Changing Properties of Channels 3553
- Pausing/Resuming 3554
- Supported Object Types for pause.htm 3554
- Error Handling, Rescanning 3555
- Rescanning, Triggering Auto-Discovery 3555
- Reordering Objects in the Sensor Tree 3555
- Report-related 3555
- Notification-related 3556
- Adding/Deleting Objects 3556
- Supported Object Types for duplicateobject.htm 3556
- Duplicating Sensors and Changing Clone Settings 3557
- Setting Geo Location 3558

Changing Object Settings

Authentication with API key or user name and passhash 3240 (or user name and password) must always be included in each PRTG API request. See section HTTP API 3513 for more information.

Examples

Rename an object:

```
/api/rename.htm?id=objectid&value=newname
```

Set priority of an object (valid values for x are 1 to 5):

```
/api/setpriority.htm?id=objectid&prio=x
```

Change properties of objects:

```
/api/setobjectproperty.htm?id=id_of_object&name=property_name&value=new_value
```

This function can change most string and number properties of objects (names, numeric values, object identifiers (OID), etc.). Use it with caution. You can discern the property_name parameter by opening the Settings page of an object and looking at the HTML source of the INPUT fields. For example, the INPUT field for the tags of an object has the name tags_. Leave away the underscore _ and use tags as a value for the property_name parameter.

Supported Object Types for rename.htm

rename.htm supports the following object types:

- group
- device
- sensor
- map
- report
- library
- notification template
- schedule
- user
- user group

rename.htm does not support other object types.

Switch Inheritance Off/On

This API call sets the inherit setting of objects (location, credentials, compatibility options, proxy settings, scanning interval, access rights, channel unit). In general, this works like changing properties for any other object.

Example

Replace the parameter inheritType with the name of the inheritance type:

```
/api/setobjectproperty.htm?id=id_of_object&name=inheritType_&value=0_or_1
```

This internal name must be followed by an underscore (_), in contrast to changing properties above. Use the value 0 for switching off inheritance, and 1 for switching on inheritance. For example, the inheritance type for the scanning interval setting has the name intervalgroup_. Thus, this specific part in the URL is &name=intervalgroup_&value=0 (switches off inheritance for scanning interval).

This call does not work with the Schedules, Dependencies, and Maintenance Window settings.

Changing Properties of Channels

With this API call, you can change a sensor's channel settings 3121. In general, this works like changing properties of any other object. To set channel properties via the PRTG API, you need to provide

- the ID of a sensor (parameter id),
- a subtype (channel for channels), and
- a subid (ID of the channel that you want to edit).

Example: Enabling and Setting Limits for Channels

Set limits for channels:

```
/api/setobjectproperty.htm?
id=sensorid&subtype=channel&subid=0&name=limitmaxerror&value=limitvalue
```

You must set the limits for a channel before you can enable limits.

Enable limits for channels:

```
/api/setobjectproperty.htm?
id=sensorid&subtype=channel&subid=0&name=limitmode&value=1
```

For example, the following API call sets the upper error limit of a channel with the ID 0 of a sensor with the ID 2970 to the value 25.

```
/api/setobjectproperty.htm?
id=2970&subtype=channel&subid=0&name=limitmaxerror&value=25
```

For Toplists, you can use the subtype toplist to change the properties. When using this subtype, subid is the ID of a Toplist.

For a list of available channel parameters, see section Single Object Property 3518.

Pausing/Resuming

Examples

Pause a sensor or object indefinitely:

```
/api/pause.htm?id=objectid&pausemsg=yourmessage&action=0
```

Pause a sensor or object for x minutes:

```
/api/pauseobjectfor.htm?id=objectid&pausemsg=yourmessage&duration=x
```

The pause message is optional. You can leave out the parameter &pausemsg=yourmessage if you do not want to display a message.

Simulate an error for a sensor:

```
/api/simulate.htm?id=objectid&action=1
```

simulate.htm only works for sensors in the Up, Warning, Unusual, or Unknown status 179.

Resume monitoring of a sensor or object:

```
/api/pause.htm?id=objectid&action=1
```

Supported Object Types for pause.htm

pause.htm supports the following object types:

- probe
- group

- device
- sensor
- notification
- user account

pause.htm does not support other object types.

Error Handling (Acknowledge Alarm)

Example

Acknowledge the Down status:

```
/api/acknowledgealarm.htm?id=objectid&ackmsg=yourmessage
```

Rescanning, Triggering Auto-Discovery

Examples

Scan a sensor now:

```
/api/scannow.htm?id=objectid
```

Run an auto-discovery for a group or device:

```
/api/discovernow.htm?id=objectid&template=filename
```

Providing a device template[3163] for auto-discovery is optional. You can leave out the parameter &template=filename to run the auto-discovery with the options defined in the object's settings. If you use a template, provide the whole file name including file extension surrounded by double quotation marks (") and encode whitespaces, if necessary. Example: &template="Linux%20SNMP.odt"

discovernow.htm overrides the Auto-Discovery setting of the target group or device. If it is set to No auto-discovery, it automatically changes to Standard auto-discovery (recommended).

Reordering Objects in the Sensor Tree

Example

Move an object in the sensor tree (x can be up, down, top, bottom):

```
/api/setposition.htm?id=objectid&newpos=x
```

Report-related

Example

Add a group, device, or sensor to a report:

```
/api/reportaddsensor.htm?id=reportid&addid=objectid
```

Notification-related

Example

Trigger a notification immediately for testing purposes:

```
/api/notificationtest.htm?id=objectid
/api/notificationtest.json?id=objectid
```

objectid is the ID of the notification template.

Adding/Deleting Objects

Adding and deleting objects in your configuration is the most complex and potentially most critical process when using the PRTG API. Keep in mind that adding or deleting objects is much better guided in the normal PRTG web interface with more warnings and alerts.

We recommend that you use the PRTG web interface for adding and deleting objects, if possible.

Deleting Objects

API calls to the delete function immediately delete the referenced object including all subobjects, if there are any. For example, deleting a group deletes all its devices and sensors. There is no way to undo a deletion, so use this function with care.

Example

Delete an object:

```
/api/deleteobject.htm?id=objectid&approve=1
```

Adding Objects

Adding completely new objects from scratch is not supported via the PRTG API because of the complexity of object creation and its parameters. To add new objects to PRTG, create a "master" object that is cloned into new objects.

Supported Object Types for duplicateobject.htm

duplicateobject.htm supports the following object types:

- group
- device
- sensor

- report
- library
- map
- notification template

duplicateobject.htm does not support other object types.

Examples

Duplicate a group:

```
/api/duplicateobject.htm?
id=id_of_group_to_clone&name=new_name&targetid=id_of_target_group
```

Duplicate a device:

```
/api/duplicateobject.htm?
id=id_of_device_to_clone&name=new_name&host=new_hostname_or_ip&targetid=id_of_targe
t_group
```

Duplicate a sensor:

```
/api/duplicateobject.htm?
id=id_of_sensor_to_clone&name=new_name&targetid=id_of_target_device
```

Duplicate a library:

```
/api/duplicateobject.htm?id=id_of_library_to_clone&name=new_name
```

Duplicate a notification template:

```
/api/duplicateobject.htm?id=id_of_notification_template_to_clone&name=new_name
```

If duplicateobject succeeds, the PRTG core server replies with a redirect to the URL of the new object (for example, /sensor.htm?id=1234), so your application should parse the new object ID from this URL.

When a group, device, or sensor is cloned, it is initially set to Paused so you have the chance to edit parameters as desired. You must resume it with an API call afterward.

The API calls for duplicating reports, maps, libraries, and notification templates do not require a targetid.

Duplicating Sensors and Changing Clone Settings

The following process duplicates a sensor, changes some settings, and then starts monitoring:

Example

Duplicate the sensor (the server replies with a redirect to the new object's web page, for example /sensor.htm?id=10214, parse id 10214 from the URL):

```
/api/duplicateobject.htm?id=2002&name=mynewsensor&targetid=2001
```

Rename the new sensor:

```
/api/setobjectproperty.htm?id=10214&name=name&value=newname
```

Change the OID (in this example for an SNMP Custom sensor):

```
/api/setobjectproperty.htm?id=10214&name=oid&value=1.2.3.4.5.6.7
```

Resume monitoring for the new sensor:

```
/api/pause.htm?id=10214&action=1
```

Setting Geo Location

You can set the location of any object via an API call. Provide the object ID together with parameters for location and/or longitude and latitude.

If only the location parameter is specified, the PRTG core server executes the geo location lookup (this can take up to three minutes). Provide the name of the location, for example, New York. It is shown in the Location settings, no matter the longitude or latitude.

If the longitude and latitude parameter is specified, the marker in the map is set to this position, no matter of the location parameter. Provide longitude and latitude separated by a comma, for example -73.998672,40.714728.

Example

Set the geo location of an object:

```
/api/setlonlat.htm?
id=objectid&location=name_of_object_location&lonlat=longitude,latitude
```

More

KNOWLEDGE BASE

How can I use the PRTG Application Programming Interface (API)?

- https://kb.paessler.com/en/topic/593

14.2.6 Custom Sensors

Custom sensors can perform a number of monitoring tasks that extend the standard sensor set. Apart from parameterized versions of Simple Network Management Protocol (SNMP), Packet Sniffer, and NetFlow sensors, you can create your own sensors using Windows Management Instrumentation Query Language (WQL) or Python, by compiling an .exe file, using any Windows software development tool, and you can request any Representational State Transfer (REST) application programming interface (API) that returns JavaScript Object Notation (JSON) or Extensible Markup Language (XML) and map the results to channels.

The following documentation describes the custom EXE/Script 3559, Python Script, and SSH Script sensors. The defined XML and JSON formats for the advanced sensors are also used for advanced HTTP data sensors and the REST Custom sensor.

For more information about custom sensors based on SNMP, Windows Management Instrumentation (WMI), Packet Sniffing, and Flow (NetFlow, jFlow, sFlow, IPFIX), see the respective custom sensors 3116.

For each sensor interval, PRTG can run an external process. The process can be a Windows .exe file, or a .bat, .cmd, .vbs, or PowerShell file, as well as a Python or Secure Shell (SSH) script.

In this section:

- Standard and Advanced EXE/Script Sensor 3559
- Standard and Advanced SSH Script Sensor 3560
- Interface Definition for EXE/BAT/CMD/VBS/PowerShell/SSH Sensors 3560
- Return Values for EXE/BAT/CMD/VBS/PowerShell/SSH Sensors 3561
- Standard EXE/Script Sensor 3561
- SSH Script Sensor 3562
- Advanced Script, HTTP Data, and REST Custom Sensors 3562
- Advanced Script, HTTP Data, and REST Custom Sensors: Elements 3564
- Command-line Parameters 3570
- Escape Special Characters and Whitespaces in Parameters 3572
- Environment Values 3572

Standard and Advanced EXE/Script Sensor

You must create the sensor as a file and store it in a specific subfolder on the probe system. In a cluster, you must store it on each cluster node.

Place executables (.exe), batch files (.cmd, .bat), VBS scripts (.vbs), or PowerShell scripts (.ps1) into a subfolder of the PRTG program directory 3636. For the standard EXE/Script sensor, this is the following subfolder of the PRTG program directory:

```
Custom Sensors\EXE
```

If your executable or script returns XML or JSON, you use it with the EXE/Script Advanced sensor. In this case, store your file in the following subfolder of the PRTG program directory:

```
Custom Sensors\EXEXML
```

You find a sample set of demo sensors in these subfolders, too. As soon as a file is placed into the subfolders mentioned above, you can create your own custom EXE sensor and select the new file from the list of files.

The probe then executes the file on the probe system using the account configured for the PRTG probe service (the default is system). The local probe runs the file on the local PRTG core server system. For remote probes, the file actually runs on the remote probe system.

If you use a PowerShell script (.ps1) and if the PowerShell Security Enhancement experimental feature 3314 is enabled, scripts that use the write-host cmdlet to provide their output to PRTG do not work. Use the write-output cmdlet instead.

We recommend that you not edit the demo files. Create your own new files and make sure to give them unique names that do not start with Demo, for example.

If your custom sensor code relies on other files (for example, .NET framework, Windows PowerShell) you must manually copy or install these files on the probe system.

EXE sensors fail if they attempt to open any graphical user interface windows using the Win32 APIs. This is not allowed for processes that are started by a system service.

Standard and Advanced SSH Script Sensor

You must create the sensor as an SSH script and place it in a specific directory on the target system running your Linux/Unix installation where the script is executed.

Place your SSH script files for the standard SSH Script sensor in the following directory of the target system:

```
/var/prtg/scripts
```

If your SSH script returns XML or JSON, you use it with the SSH Script Advanced sensor. In this case, store your file in the following directory of the target system:

```
/var/prtg/scriptsxml
```

As soon as a file is placed into the respective directory, you can create your own SSH script sensor and select the new script file from the list of scripts.

With each scanning interval, PRTG executes the script on the target system and receives the result as a sensor result.

Interface Definition for EXE/BAT/CMD/VBS/PowerShell/SSH Sensors

Every time the sensor is run, the selected file is executed. The string entered in the Parameters field of the sensor's settings is used as command line (you can use placeholders, see Command-line Parameters 3570). The executable file must send the results to the Standard OUT. For the format of returned data, see below.

If the executable file does not return control to the PRTG process, it is killed as soon as the timeout value set for this sensor is reached.

You can test the .exe file that you want to use for the sensor via the command line (cmd.exe). To do so, start the .exe file and pipe the results into a file.

Example

```
sensorexe parameter > result.txt
```

The results are written into the file result.txt and you can check the results with notepad or any other text editor.

Remarks

- For PowerShell scripts, make sure that they are executed by either signing the files or changing the security policy for Powershell.exe accordingly.
- In SSH scripts, you can use alphanumeric characters and the special characters ".", "_", "-", "=", and "/" outside of quoted strings in the Parameters field of the sensor's settings.
- The API interface for custom EXE sensors is compatible with the custom EXE sensors provided by PRTG.

Return Values for EXE/BAT/CMD/VBS/PowerShell/SSH Sensors

The expected return values are different, depending on the type of EXE/Script sensor used. The standard sensor needs a simple value:message pair. The EXE/Script Advanced sensor processes an XML or JSON return value. When using the standard SSH Script sensor, it expects returncode:value:message as result. See details below.

Standard EXE/Script Sensor

The returned data for standard EXE/Script sensors must be in the following format:

```
value:message
```

Value has to be a 64-bit integer or float. It is used as the resulting value for this sensor (for example, bytes, milliseconds) and stored in the database. The message can be any string (maximum length: 2000 characters).

The exit code of the executable file has to be one of the following values:

Value	Description
0	OK
1	WARNING
2	System Error (for example, a network/socket error)
3	Protocol Error (for example, web server returns a 404)

Value	Description
4	Content Error (for example, a web page does not contain a required word)

SSH Script Sensor

The returned data for standard SSH Script sensors must be in the following format:

```
returncode:value:message
```

Value has to be a 64-bit integer or float. It is used as the resulting value for this sensor (for example, bytes, milliseconds) and stored in the database. The message can be any string (maximum length: 2000 characters).

The SSH script returncode has to be one of the following values:

Value	Description
0	OK
1	WARNING
2	System Error (for example, a network/socket error)
3	Protocol Error (for example, web server returns a 404)
4	Content Error (for example, a web page does not contain a required word)

Advanced Script, HTTP Data, and REST Custom Sensors

The returned data for the EXE/Script Advanced, Python Script Advanced, SSH Script Advanced, HTTP Push Data Advanced, HTTP Data Advanced, and HTTP IoT Push Data Advanced sensors must be in XML or JSON format, the REST configuration file for the REST Custom sensor must be available as JSON template.

Most parameters have a default value and are not required. The following minimum examples leave most parameters to their default values and return two static channel values.

Examples

XML Return Format: Minimum Example:

```
<prtg>
<result>
<channel>First channel</channel>
<value>10</value>
</result>
<result>
<channel>Second channel</channel>
<value>20</value>
</result>
</prtg>
```

To return an error, the XML format is:

```
<prtg>
<error>1</error>
<text>Your error message</text>
</prtg>
```

JSON Return Format: Minimum Example

```
{
 "prtg": {
  "result": [
   {
    "channel": "First channel",
    "value": 10
   },
   {
    "channel": "Second channel",
    "value": 20
   }
  ]
 }
}
```

To return an error, the JSON format is:

```
{
 "prtg": {
  "error": 1,
  "text": "Your error message"
 }
}
```

You can find a more detailed demo script for the EXE/Script Advanced sensor in the \Custom Sensors\EXEXML subfolder of the PRTG program directory [3636]. You find demo files for other sensors in the \Custom Sensors folder as well.

Advanced Script, HTTP Data, and REST Custom Sensors: Elements

You can optionally define the encoding of your .xml file at the beginning of the document. For example, to define UTF-8, you would use:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

You can use the following elements in the section between <result> and </result>. In each section, you can return one channel. You can define a maximum of 50 channels.

If you exceed this limit, PRTG tries to display all channels. However, be aware that this is an unsupported procedure and you experience limited usability and performance.

For XML output, the tag names are not case-sensitive. For example, you can use both "VALUE" and "value". For JSON output, the tag names are case-sensitive but you can also use lowercase. For example, you can use both "CustomUnit" and "customunit".

Tag	Mandatory	Description	Possible Content
<Channel>	Yes	Name of the channel as displayed in user interfaces. This parameter is required and must be unique for the sensor.	Any string
<Value>	Yes	The value as integer or float. Make sure the <Float> setting matches the kind of value provided. Otherwise PRTG shows 0 values.	Integer or float value
<Unit>	No	The unit of the value. The default is Custom. This is useful for PRTG to convert volumes and times.	BytesBandwidth BytesDisk Temperature Percent TimeResponse TimeSeconds Custom Count CPU: This is a % unit that is accounted to the CPU load in index graphs. BytesFile

Tag	Mandatory	Description	Possible Content
			SpeedDisk SpeedNet TimeHours
<CustomUnit>	No	If Custom is used as unit, this is the text displayed behind the value.	Any string (keep it short)
<SpeedSize> <VolumeSize>	No	Size used for the display value. For example, if you have a value of 50000 and use Kilo as size, the display is 50 kilo #. The default is One (value used as returned). For the Bytes and Speed units, this is overridden by the setting in the user interface.	One Kilo Mega Giga Tera Byte KiloByte MegaByte GigaByte TeraByte Bit KiloBit MegaBit GigaBit TeraBit
<SpeedTime>	No	See above, used when displaying the speed. The default is Second.	Second Minute Hour Day
<Mode>	No	Select if the value is an absolute value or counter. The default is Absolute.	Absolute Difference
<Float>	No	Define if the value is a float. The default is 0 (no). If set to 1 (yes), use a dot as decimal separator in values.	0 (= no, integer) 1 (= yes, float)

Tag	Mandatory	Description	Possible Content
		Define decimal places with the <DecimalMode> element.	
<DecimalMode>	No	Init value for the Decimal Places option. If 0 is used in the <Float> element (use integer), the default is Auto. Otherwise (for float) the default is All. You can change this initial setting later in the sensor's channel settings 3121.	Auto All
<Warning>	No	If enabled for at least one channel, the entire sensor is set to the Warning status. The default is 0 (no).	0 (= no) 1 (= yes)
<ShowChart>	No	Init value for the Show in graphs option. The default is 1 (yes). The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and can be omitted). You can change this initial setting later in the sensor's channel settings.	0 (= no) 1 (= yes)
<ShowTable>	No	Init value for the Show in tables option. The default is 1 (yes). The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and can be omitted). You can change this initial setting later in the sensor's channel settings.	0 (= no) 1 (= yes)
<LimitMaxError>	No	Define an upper error limit for the channel. If enabled, the sensor is set to the Down status if this value is exceeded and the LimitMode is activated.	String with numbers, surrounded by quotation marks (")

Tag	Mandatory	Description	Possible Content
		ⓘ Provide the value for the limit in the unit of the base data type as it is used in the <Value> element of this section. When a sensor shows the Down status triggered by a limit, it still receives data in its channels. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and can be omitted). You can change this initial setting later in the sensor's channel settings.	
<LimitMaxWarning>	No	Define an upper warning limit for the channel. If enabled, the sensor is set to the Warning status if this value is exceeded and the LimitMode is activated. ⓘ Provide the value for the limit in the unit of the base data type as it is used in the <Value> element of this section. When a sensor shows the Down status triggered by a limit, it still receives data in its channels. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and can be omitted). You can change this initial setting later in the sensor's channel settings.	String with numbers, surrounded by quotation marks (")
<LimitMinWarning>	No	Define a lower warning limit for the channel. If enabled, the sensor is set to the Warning status if this value falls below the defined limit and the LimitMode is activated. ⓘ Provide the value for the limit in the unit of the base data type as it is used in the <Value> element of this section. When a sensor shows the Down status triggered by a limit, it still receives data in its channels.	String with numbers, surrounded by quotation marks (")

Tag	Mandatory	Description	Possible Content
		ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and can be omitted). You can change this initial setting later in the sensor's channel settings.	
<LimitMinError>	No	Define a lower error limit for the channel. If enabled, the sensor is set to the Down status if this value falls below the defined limit and the LimitMode is activated. ⓘ Provide the value for the limit in the unit of the base data type as it is used in the <Value> element of this section. When a sensor shows the Down status triggered by a limit, it still receives data in its channels. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and can be omitted). You can change this initial setting later in the sensor's channel settings.	String with numbers, surrounded by quotation marks (")
<LimitErrorMsg>	No	Define an additional message. It is added to the sensor's message when entering the Down status that is triggered by a limit. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and can be omitted). You can change this initial setting later in the sensor's channel settings.	Any string
<LimitWarningMsg>	No	Define an additional message. It is added to the sensor's message when entering the Warning status that is triggered by a limit.	Any string

Tag	Mandatory	Description	Possible Content
		The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and can be omitted). You can change this initial setting later in the sensor's channel settings.	
<LimitMode>	No	Define if the limit settings defined above are active. The default is 0 (no; limits inactive). If 0 is used, the limits are written to the channel settings as predefined values, but limits are disabled. The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and can be omitted). You can change this initial setting later in the sensor's channel settings.	0 (= no) 1 (= yes)
<ValueLookup>	No	Define if you want to use a lookup file (for example, to view integers as status texts). Enter the ID of the lookup file that you want to use, or omit this element to not use lookups. The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and can be omitted). You can change this initial setting later in the sensor's channel settings.	Any string
<NotifyChanged>	No	If a returned channel contains this tag, it triggers a change notification that you can use with the change trigger 3252 to send a notification.	No content required

You can use the following elements in the section between <prtg> and </prtg>, outside the <result> section.

For XML output, the tag names are not case-sensitive. For example, you can use both "TEXT" and "text". For JSON output, the tag names are case-sensitive but you can also use lowercase. For example, you can use both "Text" and "text".

Tag (Case Insensitive)	Mandatory	Description	Possible Content
<Text>	No	Text the sensor returns in the Message field with every scanning interval. There can be one message per sensor, regardless of the number of channels. The default message is OK. This element has to be provided outside of the <result> element.	Any string Maximum length: 2000 characters The number sign (#) is not supported in sensor messages. If a message contains a number sign, the message is clipped at this point.
<Error>	No	If enabled, the sensor returns the Down status. This element can be combined with the <Text> element to show an error message. The default is 0. This element has to be provided outside of the <result> element. A sensor in this error status cannot return any data in its channels. If used, all channel values in the <result> section are ignored.	0 (= no) 1 (= yes, set sensor to error; ignore <result> section)

Each run (sensor scan) might return either any number of channels (<result>...</result>) or one error response. It is not possible to mix result and error entries.

You can either write the XML output to standard OUT line by line, or give back the entire expression in one line without breaks.

Command-line Parameters

In the parameter field, you can use the following placeholders:

Placeholder	Description
%sensorid	The ID of the EXE/Script sensor.

Placeholder	Description
%deviceid	The ID of the device the sensor is created on.
%groupid	The ID of the group the sensor is created in.
%probeid	The ID of the probe the sensor is created on.
%host	The IP address/DNS name of the device the sensor is created on.
%device	The name of the device the sensor is created on.
%group	The name of the group the sensor is created in.
%probe	The name of the probe the sensor is created on.
%name	The name of the EXE/Script sensor.
%windowsdomain	The domain for Windows access (can be inherited from parent).
%windowsuser	The user name for Windows access (can be inherited from parent).
%windowspassword	The password for Windows access (can be inherited from parent).
%linuxuser	The user name for Linux access (can be inherited from parent).
%linuxpassword	The password for Linux access (can be inherited from parent).
%snmpcommunity	The community string for SNMP v1 or v2 (can be inherited from parent).

You need to escape placeholders that you use in the parameter field with quotation marks so that they can be correctly resolved from the command line.

You need to escape special characters and whitespaces in your parameters and surround them with double quotation marks ("). See section Escape Special Characters and Whitespaces in Parameters 3559 for details.

In SSH scripts, you can use alphanumeric characters and the special characters ".", "_", "-", "=", and "/" outside of quoted strings.

See section Inheritance of Settings 135 for more information on inherited settings.

Escape Special Characters and Whitespaces in Parameters

You need to escape special characters in parameters that you pass to an executable or script and surround them with quotation marks to make sure that the characters are correctly interpreted. PowerShell scripts in particular require adequate escaping so that the parameters are passed in a valid PowerShell syntax. PRTG automatically does most of the escaping for you.

Follow these rules to escape special characters and whitespaces in the parameters fields:

- Use double (") or single (') quotation marks for parameters that contain whitespaces.

```
-name "Mr John Q Public"
-name 'Mr John Q Public'
```

- Use double quotation marks (") for parameters that already contain single quotation marks (').

```
-name "Mr 'John Q' Public"
```

- Use single quotation marks (') for parameters that already contain double quotation marks (").

```
-name 'Mr "John Q" Public'
```

- Use a backslash (\) to escape and pass a literal double quotation mark.

```
-name pub\"lic
```

- Use double quotation marks (") for parameters that contain double (") and single (') quotation marks and escape double quotation marks (").

```
-name "pu'b\"lic"
```

ⓘ In SSH scripts, you can use alphanumeric characters and the special characters ".", "_", "-", "=", and "/" outside of quoted strings.

ⓘ We recommend that you do not pass passwords in parameters. Use placeholders instead. See section Custom Sensors 3572 for details.

Environment Values

If the Set placeholders as environment values option is enabled in the sensor's settings, the values of all placeholders available for command-line parameters 3570 are additionally provided as "Environment Variables" during run time, so you can use them in your executable or script file. The variables' names are the same as for placeholders mentioned above, with the prefix prtg_ and without the % character. For example, refer to the sensor's own name by using the variable prtg_name.

Additionally, the following variables are available:

Variable	Description
prtg_version	The version number of your PRTG installation.
prtg_url	The IP address/DNS name of your PRTG installation.
prtg_primarychannel	The ID of the sensor's primary channel (1 if not set).

More

You can find sample projects for these custom sensors and more information about custom scripts here:

- \Custom Sensors\EXE subfolder of the PRTG program directory 3636.

KNOWLEDGE BASE

Custom sensors

- https://kb.paessler.com/en/tags/custom-script-exe

Guide for PowerShell-based custom sensors

- https://kb.paessler.com/en/topic/71356

PAESSLER WEBSITE

You can find scripts for custom sensors that were written by dedicated PRTG customers in the PRTG Sensor Hub.

- https://www.paessler.com/sensor-hub

14.2.7 Custom Notifications

In addition to the various standard methods for notifications, you can define your own notifications that can trigger desired actions. The following documentation describes these custom notifications. You can also combine different notification methods in one notification.

For more general information about notifications based on email, messaging, and others, see section Notifications 3173.

Execute HTTP Action

This notification method executes a GET request or sends any POST, PUT, or PATCH data to a custom URL. You can execute specific actions on a web server or control any web service that accepts commands via one-time HTTP requests. Whenever a notification of this kind is triggered, the HTTP action is sent.

With this method, you can also call any application programming interface (API) function of the PRTG web interface. For example, you can automatically pause a sensor or acknowledge an alarm.

Authentication with API key or user name and passhash 3240 (or user name and password) must always be included in each PRTG API request. See section HTTP API 3513 for more information.

Examples

To automatically pause the sensor that triggers the notification, enter the following HTTP action:

```
http://yourserver/api/pause.htm?id=%
sensorid&action=0&username=myuser&password=mypassword
```

To use the notification to automatically acknowledge the alarm that triggered it, enter this HTTP action:

```
http://yourserver/api/acknowledgealarm.htm?id=%sensorid&ackmsg=Auto-
Acknowledged&username=myuser&password=mypassword
```

For more information about authentication within the URL and for other possible actions you can configure, see sections HTTP API 3512 and Object Manipulation 3552.

Execute Program

With this notification method, you can execute a script or a program as an external process. It can be a Windows executable file or a .bat, .cmd, or PowerShell file. You can use .exe, .com, .bat, .cmd, .vbs, or .ps1 files.

You must create the notification as a file and place it in a specific subfolder on the PRTG core server system (in a cluster, copy the files to every cluster node).

Place executables (.exe, .com), batch files (.cmd, .bat), VBS scripts (.vbs), or PowerShell scripts (.ps1) into the folder:

```
\Notifications\EXE
```

As soon as a file is placed into the subfolder, you can create or edit your own custom execute program notification and select the new file from the list of files. You can also enter start parameters and use PRTG placeholders for this.

Notes

- PRTG executes the file on the local PRTG core server system using the account configured for the PRTG core server service (the default is system).
- If your custom notification's code relies on other files (for example, .dll, .NET framework, or Windows PowerShell), you must copy/install these files on the PRTG core server system manually.
- Make sure the return code of the executable is 0 (zero). Otherwise PRTG assumes something went wrong with the notification and tries to send it up to 3 times.
- If you run PRTG in a cluster, copy the respective files to every cluster node to make sure the notification also works when the primary master node is not reachable.
- EXE notifications fail if they attempt to open any graphical user interface windows using the Win32 APIs (this is not allowed for processes that are started by a system service).
- To remotely run PowerShell scripts, make sure that you set the according Execution Policy. For more information, see the Knowledge Base: PowerShell 32 bit and 64 bit and Execution Policy.

Placeholders

For more information about the placeholders you can use, see section List of Placeholders for Notifications 3776.

PRTG Sensor Hub

You can find scripts for custom sensors that were written by dedicated PRTG customers in the PRTG Sensor Hub.

More

KNOWLEDGE BASE

Custom notifications

- https://kb.paessler.com/en/tags/custom-notification

PowerShell 32 bit and 64 bit and Execution Policy

- https://kb.paessler.com/en/topic/20443

14.2.8 Mini Probe API

Important Notice

We do not further develop the Mini Probe API because we plan major changes to the underlying PRTG Application Programming Interface (PRTG API). You can still use the Mini Probe API "as is" but note that it may be deprecated at any time.

Knowledge Base: Where can I find PRTG mini probes which are ready to use?

Mini probes allow users to create small probes on any device to meet specific needs. In general, probes are the part of PRTG that run monitoring processes and deliver monitoring results back to the PRTG core server. Mini probes gather monitoring data from platforms where it is not possible or is inapplicable to use the common local and remote probes of PRTG. Mini probes have a less complex implementation than standard probes so that you can create them on any platform. The only requirement is HTTPS connectivity to send monitoring data to your PRTG core server.

With the current version of PRTG, you can use the mini probe interface with your custom code to implement solutions to special scenarios that you might have in your network. Note that there are major changes planned to the underlying PRTG API. Therefore, any code you write now likely needs to be changed later, so it can be used for future versions of PRTG. For example, if the available HTTP Push sensors are not sufficient for your needs, you can still use the Mini Probe API.

Because the mini probe requires a Secure Sockets Layer (SSL) secured connection to the PRTG core server, it is not possible by default to connect if SSL is deactivated for PRTG. This is necessary because probably unencrypted passwords are transferred between the probe and the PRTG core server. So it is important to encrypt the connection even on internal routes. If your network setup ensures security in a different way (for example, a VPN), you can use a registry key option for disabling SSL to get a connection to your mini probe.
For more details, see the Knowledge Base: How can I disable SSL for Mini Probes?

The Mini Probe API is not available in PRTG Hosted Monitor

In this section:

- Differences Between Probe Types 3577
- The PRTG Mobile Probe Protocol 3578
  - Communication and Security 3578
  - Authentication 3578
  - HTTP Requests 3578
- JSON Definition 3580
  - Sensor Definition 3580
  - Tasks Definition 3585
  - Data Definition 3587

Differences Between Probe Types

The following table shows technical specifications of the two probe types in PRTG.

Functionality	Local and Remote Probes	Mini Probe
Connection Protocol from Probe to PRTG Core Server	Protocol from PRTG	HTTPS
Security	Data is secured with SSL and an access key. New probe connections must be approved by an administrator. IP address and globally unique identifier (GUID) filtering is possible.	The same security level as for local and remote probes.
Estimated Limit for Sensors per Probe	Several thousand sensors	Fewer than 100 sensors
Estimated Limit for Probes per Installation	Hundreds of probes	Fewer than 100 probes
Estimated Minimum Scanning Interval	Some seconds	At least 60 seconds
Estimated Number of Sensors	More than 200	A few
Updates to New PRTG Versions by the PRTG Core Server	Yes	No
Complexity	High	Very low
Documented API	No	Yes
Supported Platforms	At least Windows (32-bit/64-bit)	Any platform
Scheduling of Sensor Requests Performed by	Probe	Probe
Code Managed by	Paessler AG	Writer of the probe
Probe Scans for Available Measurements Beforehand	Yes	No
Sensors Support Inheritance of Settings 135	Yes	No

Functionality	Local and Remote Probes	Mini Probe
Limitations by Administrator for Allowed Sensors that a User Can Create	Yes	No

The PRTG Mobile Probe Protocol

The PRTG Mini Probe Protocol (PMPP) is a simple, lightweight protocol that extends PRTG with custom remote probes. It can be implemented in a variety of programming languages and runs on any kind of platform. These include, for example, Linux, Android, macOS, and iOS.

Mini probes are not intended for high performance monitoring and support only the sensors that you implement.

Communication and Security

The PMPP uses GET and POST requests via HTTPS to communicate with the PRTG core server. All requests are sent to the defined Transmission Control Protocol (TCP) port for the common PRTG web server (the default port is 443 for SSL), or you specify an extra port explicitly for mini probe connections in the Core & Probes 3325 settings. Mini probes use the GET method to receive tasks, and the POST method to send information about the probe and the monitoring results to the PRTG core server.

The data format of some HTTP fields has to be JavaScript Object Notation (JSON) encoded. See JSON Definition 3580 for data definitions that require JSON. All exchanged data is secured with SSL.

Authentication

The PMPP uses the same authentication methods as the common remote probes in PRTG. The authentication includes the following steps:

- Allow and deny IP addresses filter
- Deny global ID (GID) filter
- Access key
- Unique GID that must be approved in the PRTG web interface
- In addition, mini probes must be allowed to connect in the probe connection settings in PRTG. Additionally, you must provide the mini probe's IP address in the Allow IP Addresses field (or enter any).

There are no sessions on the server: Every request must contain the required authentication information.

For more details, see section Core & Probes 3325.

HTTP Requests

The PMPP includes three different HTTP requests that are sent to the PRTG web server:

- announce: An announce request is sent once when the probe starts. Afterward, the task and data commands are run in a scheduled manner.
- tasks: With a tasks request, the probe requests a list of tasks to perform.
- data: The data request sends the monitoring results to the core.

All requests of the mini probe to the PRTG core server must contain the following HTTP fields:

- gid: The unique GID of the probe. We recommend a GUID that is generated by the operating system. This identifier must stay the same for as long as the probe installation exists. You can use any string. If you clone a probe, you must update this field to a new value.
- key: An access key as defined in the probe settings of the PRTG core server. The key has to be encoded in SHA1 hash (for example, key=a94a8fe5ccb19ba61c4c0873d391e987982fbbd3).
- protocol: The version of the protocol you use. Currently, this value is "1"

All requests return common HTTP response codes.

For more information, see section HTTP API 3512.

Announce Request

The announce request uses the POST method and provides all required information about the capabilities of the mini probe for the PRTG core server. The target URL is https://<yourPRTGserver>/probe/announce.

This HTTP request must be sent at least once to be able to add sensors. We recommend you send this request every time the probe starts. You should NOT send it with every scanning interval.

The announce request must contain the following HTTP fields:

- name: The name of the mini probe. PRTG uses this name to create a corresponding node in the device tree.
- version: The version number of the mini probe you have implemented. This is a single integer, for example, 1.
- baseinterval: The number of seconds between two calls of the task/data requests. We recommend 60 or 300 seconds. Depending on the usage, higher or lower values are possible.
- sensors: The definition of supported sensors in JSON format.
  See Sensor Definition 3580 for more information.
- icon (optional): You can optionally send the file name of a device icon to show it for the mini probe device in the PRTG web interface.

If you change the definition of a sensor that has already been announced, these changes are only active after the next start of PRTG. A definition never changes while PRTG is running. However, there is one exception: Setting the "deprecated" flag works without any restart. Because of this, you can replace a sensor with a new one that uses a different definition.

Tasks Request

The tasks request uses the GET method and is sent from your mini probe in the defined scheduler interval to the PRTG core server (for example, every 5 minutes). The target URL is

```
https://<PRTGserver>/probe/tasks
```

This HTTP request returns a list of tasks in JSON format that need to be run by the mini probe.

See Tasks Definition 3585 for more information.

Data Request

The data request uses the POST method and contains the HTTP field data. This HTTP field contains any number of sensor results in JSON format. The target URL is

```
https://<PRTGserver>/probe/data
```

You can split the results of one tasks list into several result requests (for example, if some sensors are faster than others). The mini probe should combine as many results as possible into one request but keep the time between measurement and reporting of the value at a low level.

See Data Definition 3587 for more information.

JSON Definition

All data definitions of sensors, tasks, and result data of mini probes are JSON encoded. JSON is a language-independent data format that is used to transmit data objects consisting of attribute-value pairs between a server and an application. Refer to the JSON documentation for a general overview of this data format.

Sensor Definition

This section shows how you can define the available sensors for your mini probe. Sensor definitions are specified in the HTTP field "sensors" of the announce request. The sensor types definition is a JSON array where each sensor type is defined in one array element as a JSON object. A JSON object denoting a sensor definition consists of the following JSON name/value pairs:

Name	Mandatory	Description	Possible Value
kind	X	Unique identifier for the sensor type in the mini probe. Used in the tasks definition to identify the sensor type. Underscore "_" is not allowed here.	Any string
name	X	The display name of the sensor.	Any string
deprecated	—	You can flag the sensor to status deprecated. A flagged sensor can still run but this kind of sensor is not shown when you add new sensors to the probe.	1 (= deprecated) 0 (= not deprecated)

Name	Mandatory	Description	Possible Value
description	—	A short description of the sensor that is shown in the Add Sensor 413 dialog in the PRTG web interface.	Any string
help	—	A help text that is shown in a popup in the Add Sensor dialog in the PRTG web interface.	Any string
tag	—	A default tag for the sensor that is automatically added to the sensor.	Any string
default	—	A sensor of this type is automatically created with the probe if set to "default".	(= set to default) 0 (= not default)
groups	—	In the "groups" array, available settings for this sensor type are defined.	An array of grouped settings JSON objects. ⓘ See Definition of Setting Groups Objects 3581 for more information.

Definition of Setting Groups Objects

One settings group definition (one element of the "groups" array) consists of three elements:

Name	Mandatory	Description	Possible Value
name	X	The internal name of the settings group.	Any unique string
caption	X	The label of the settings group as shown in the PRTG web interface.	Any string
fields	X	The available settings of the settings group.	An array of field definition objects. ⓘ See Parameters for Setting Fields 3581 for more information.

Parameters for Setting Fields

The following table shows available JSON name/value pairs for setting fields:

Name	Mandatory	Description	Available in Type	Possible Value
type	X	Type of the field. This defines the possible content.	All	Edit Password Integer Radio ⓘ See Definition of Setting Fields: Field Types 3583 for more information.
name	X	The internal name of the field. The name has to be unique per sensor. It is sent with the settings of the probe in the task request.	All	Any unique string
caption	X	The label of the field. It is displayed left of the field.	All	Any string
required	—	If a field is defined as required, this field has to be set when adding or editing the sensor settings. The default is not required.	All	0 (= not required) 1 (= required)
default	—	The default value of the field.	All	Any string or integer (depending on the field type)
help	—	A help text that is displayed right of the field. ⓘ You can use limited BBCode: "[b]" and "[/b]" for bold, "[i]" and "[/i]" for italics, and "[br]" for line break.	All	Any string

Name	Mandatory	Description	Available in Type	Possible Value
maximum	—	The maximum value that is allowed for this field.	Integer	Integer
minimum	—	The minimum value that is allowed for this field.	Integer	Integer
options	—	A JSON array that provides several radio buttons to choose a desired option.	Radio	"name":"value" pairs. See Example 3583 below.

Example

"name":"value" pairs that define radio button options:

```
{
       "1":"This is option 1",
       "2":"This is option 2",
       "XYZ":"Another option"
}
```

Definition of Setting Fields: Field Types

A sensor type can have any number of setting fields that are organized in groups of settings. One field is one element in the "fields" array of a settings group. Currently, mini probes support four different field types for settings:

- edit: One line edit field.
- password: An edit field with masked characters.
- integer: A number field with optional minimum/maximum selection.
- radio: A selection of multiple options with radio buttons.

Example

The following is a detailed example that shows the JSON object definition of a sensor type that is used in the HTTP field sensors of the announce request.

This sensor type is called Sample Sensor and is from the type Sample. It has a description, a help text, and a default tag. There are two setting groups, Group and group2, with several setting fields (six in the first group, one in the second group). The example also shows how you can use the available JSON name/value pairs in the fields array object.

```
[
	{
	"kind":"Sample",
	"name":"Sample Sensor",
	"description":"This is a sample demo sensor",
	"help":"This is the help text of the demo sensors",
	"tag":"demosensor",
	"groups":[
		{
		"name":"Group",
		"caption":"Group",
		"fields":[
		   {
		       "type":"edit",
		       "name":"simpleedit",
		       "caption":"Edit Field",
		       },
		   {
		       "type":"edit",
		       "name":"extendededit",
		       "caption":"Edit Field 2",
		       "required":"yes",
		       "default":"Default Value",
		       "help":"Help text displayed to the right of the field"
		       },
		   {
		       "type":"integer",
		       "name":"simplenumber",
		       "caption":"Number",
		       },
		   {
		       "type":"integer",
		       "name":"number2",
		       "caption":"Number 2",
		       "required":"1",
		       "minimum":23,
		       "maximum":99,
		       "help":"Number field with limit 23-99"
		       },
		   {
		       "type":"password",
		       "name":"password",
```

```
                    "caption":"Password",
                    "help":"This is a password field"
                    },
                {
                    "type":"radio",
                    "name":"radiotest",
                    "caption":"Radio test",
                    "help":"This is a radio selection field",
                    "options":{
                                        "1":"This is option 1",
                                        "2":"This is option 2",
                                        "3":"This is option 3"
                                        },
                    "default":"2"
                    },
                    ]
            },
            {
            "name":"group2",
            "caption":"Group 2",
            "fields":[
                      {
                          "name":"testfield2",
                          "caption":"Test2",
                          "type":"edit"
                          }
                          ]
            }
        ]
      }
]
```

Tasks Definition

A tasks definition is a JSON array where each task is one object. Tasks contain all name/value pairs as defined in the sensor settings definition, which are filled with the values you have provided. Additionally, the following information is included:

Name	Mandatory	Description	Possible Value
kind	X	The type of the sensor.	String

Name	Mandatory	Description	Possible Value
sensorid	X	The ID of the sensor.	Integer
host	X	The IP address/DNS name of the parent device as specified for this device. For the probe device, the default is 127.0.0.1.	IP address/DNS name
all defined fields	X	All fields that are defined in the sensor setting group objects are included in the tasks definition as name/value pairs.	name/value pairs

ⓘ This data comes from PRTG, so the mandatory JSON objects are included automatically.

Example

Definition of two tasks, the first one is the simplest possible one without any values, and the second one uses the sensor settings objects as defined above:

```
[
	{
	"sensorid":"2009",
	"kind":"ping",
	"host":"www.google.com"
	}
	{
	"sensorid":"2010",
	"kind":"sample",
	"host":"www.paessler.com",
	"simpleedit":"Test2",
	"extendededit":"Extended Test",
	"simplenumber":"3",
	"number2":"42",
	"password":"masked text",
	"radiotest":"1",
	"testfield2":"This is test 3"
	}
]
```

Data Definition

A data definition is a JSON array where each result of a task is one object. Every array element contains the following name/value pairs:

Name	Mandatory	Description	Possible Value
sensorid	X	The ID of a specific sensor.	Integer
time	—	The time of measurement in Coordinated Universal Time (UTC)/GMT time zone as a JSON number in the Unix time format (in milliseconds since Unix epoch, which is 00:00:00 UTC on January 1, 1970). Time values must be strictly chronological, so the Unix time of each measurement must be greater than the one before. The time values should be close to the current time (which is now) to prevent sensors in the Unknown status 179. ⓘ If no time value is provided, the current time (now) is used.	JSON number defining Unix time
message	—	An optional text message.	Any string
channel	X	The channel result values.	An array of name/value pairs. ⓘ See Parameters for Data Definitions: Channel Result Values 3588 for more information.

Examples

Data definition object with sensor status OK:

```
[
   {
       "sensorid":"2003",
       "message":"Optional Message",
       "channel":[
              {
              "name":"Time",
              "mode":"integer",
              "unit":"TimeResponse",
              "value":6
              "showchart":1
              "showtable":1
              }
              {
              "name":"Pages",
              "mode":"counter",
              "unit":"Custom",
              "customunit":"Pages",
              "value":99
              }
         ]
       }
]
```

Data definition object with sensor status error:

```
[
       {
       "sensorid":"2003",
       "error":"Response",
       "code":10,
       "message":"Error Message"
       }
]
```

Parameters for Data Definitions: Channel Result Values

The following table shows name/value pairs that can be used in the "channel" array objects of data definition objects:

Name	Mandatory	Description	Possible Value
Name	X	The name of the channel as displayed in user interfaces.	Any string
Value	X	Any number without quotation marks.	An integer, float, or counter value
Mode	—	The type of the value. ⓘ Make sure that it matches the provided value, otherwise PRTG shows 0 values.	Integer, float, or counter
Unit	—	The unit of the value. ⓘ If you set the correct unit type instead of using custom units, PRTG can display received values better.	BytesBandwidth BytesMemory BytesDisk BytesFile TimeResponse TimeSeconds TimeHours Temperature Percent Count CPU: This is a % unit that is accounted to the CPU load in index graphs. Custom (define the name of the unit using the additional field customunit)
ShowChart	—	Init value for the Show in graphs option. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and may be omitted). You can change this initial setting later in the sensor's channel settings.	0 (= do not show graph) 1 (= show graph)

Name	Mandatory	Description	Possible Value
ShowTable	—	Init value for the Show in graphs option. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and may be omitted). You can change this initial setting later in the sensor's channel settings.	0 (= do not show table) 1 (= show table)
SpeedSize VolumeSize	—	Size used for the display value. For example, if you have a value of 50000 and use Kilo as size, the display is 50 kilo #. The default is One (value used as returned). ⓘ For the Bytes and Speed units, this is overridden by the setting in the user interface.	One Kilo Mega Giga Tera Byte KiloByte MegaByte GigaByte TeraByte Bit KiloBit MegaBit GigaBit TeraBit
SpeedTime	—	See above, used when displaying the speed. The default is Second.	Second Minute Hour Day

Name	Mandatory	Description	Possible Value
decimalMode	—	Init value for the Decimal Places option. If 0 is used in the float mode (use integer), the default is Automatic. Otherwise (for float), the default is All. ⓘ You can change this initial setting later in the sensor's channel settings 3121.	Automatic All Custom
decimalDigits	—	If you define Custom as decimalMode, specify the number of digits after the delimiter.	Integer
ValueLookup	—	Define if you want to use a lookup file (for example, to view integers as status texts). Enter the ID of the lookup file that you want to use, or omit this element to not use lookups. ■ See section Define Lookups 3604 for more information. ⓘ This setting is only considered on the first sensor scan, when the channel is newly created. It is ignored on all further sensor scans (and may be omitted). You can change this initial setting later in the sensor's channel settings.	Any string
LimitMaxError	—	Define an upper error limit for the channel. If enabled, the sensor is set to the Down status if this value is exceeded and the LimitMode is activated. ⓘ Provide the value for the limit in the unit of the base data type as it is used in the <Value> element of this section. When a sensor shows the Down status triggered by a limit, it still receives data in its channels.	Integer

Name	Mandatory	Description	Possible Value
		ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and may be omitted). You can change this initial setting later in the sensor's channel settings.	
LimitMaxWarning	—	Define an upper warning limit for the channel. If enabled, the sensor is set to the Warning status if this value is exceeded and the LimitMode is activated. ⓘ Provide the value for the limit in the unit of the base data type as it is used in the <Value> element of this section. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and may be omitted). You can change this initial setting later in the sensor's channel settings.	Integer
LimitMinWarning	—	Define a lower warning limit for the channel. If enabled, the sensor is set to the Warning status if this value falls below the defined limit and the LimitMode is activated. ⓘ Provide the value for the limit in the unit of the base data type as it is used in the <Value> element of this section.	Integer

Name	Mandatory	Description	Possible Value
		ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and may be omitted). You can change this initial setting later in the sensor's channel settings.	
LimitMinError	—	Define a lower error limit for the channel. If enabled, the sensor is set to the Down status if this value falls below the defined limit and the LimitMode is activated. ⓘ Provide the value for the limit in the unit of the base data type as it is used in the <Value> element of this section. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and may be omitted). You can change this initial setting later in the sensor's channel settings.	Integer
LimitErrorMsg	—	Define an additional message. It is added to the sensor's message when entering the Down status that is triggered by a limit. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and may be omitted). You can change this initial setting later in the sensor's channel settings.	Any string

Name	Mandatory	Description	Possible Value
LimitWarningMsg	—	Define an additional message. It is added to the sensor's message when entering the Warning status that is triggered by a limit. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and may be omitted). You can change this initial setting later in the sensor's channel settings.	Any string
LimitMode	—	Define if the limit settings defined above are active. The default is 0 (no; limits inactive). If 0 is used, the limits are written to the channel settings as predefined values, but limits are disabled. ⓘ The values defined with this element are only considered during the first sensor scan when the channel is newly created. They are ignored on all further sensor scans (and may be omitted). You can change this initial setting later in the sensor's channel settings.	0 (= no) 1 (= yes)
Warning	—	If enabled for at least one channel, the entire sensor is set to the Warning status. The default is 0 (no).	0 (= no) 1 (= yes)
Message	—	Text the sensor returns in the Message field with every scanning interval. There can be one message per sensor, regardless of the number of channels. The default message is OK.	Any string

Name	Mandatory	Description	Possible Value
Error	—	The type of error. ⓘ The type is not necessarily shown in PRTG.	Data: The monitored device returned a value but the sensor could not process it. Response: The monitored device reported an error. This includes timeouts, HTTP response codes, etc. Exception: Error in sensor handling. Socket: Socket error.
Code	—	The error code that is stored in the database.	Integer

More

Knowledge Base

How can I disable SSL for Mini Probes?

- https://kb.paessler.com/en/topic/60356

Where can I find PRTG mini probes which are ready to use?

- https://kb.paessler.com/en/topic/61215

14.3 Filter Rules for Flow, IPFIX, and Packet Sniffer Sensors

You can use filter rules for the Include Filter, Exclude Filter, and Channel Definition fields of packet sniffer 3439, flow, and IPFIX 3441 sensors. The filter rules are based on the following format:

```
field[filter]
```

In this section:

- Valid Fields for All Sensors 3596
- Additional Fields for Packet Sniffer Sensors Only 3597
- Additional Fields for NetFlow v5 and jFlow v5 Sensors Only 3597
- Additional Fields for NetFlow v9 and IPFIX Sensors Only 3598
- Additional Fields for sFlow Sensors Only 3599
- Valid Data Formats 3599
- Examples 3599

Valid Fields for All Sensors

Field	Possible Filter Values
IP	IP address or Domain Name System (DNS) name For more information, see section Valid Data Formats 3599.
Port	Any number
SourceIP	IP address or DNS name
SourcePort	Any number
DestinationIP	IP address or DNS name
DestinationPort	Any number
Protocol	Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP), Open Shortest Path First (OSPF), any number
ToS	Type of Service (ToS): any number
DSCP	Differentiated Services Code Point (DSCP): any number

Additional Fields for Packet Sniffer Sensors Only

Field	Possible Filter Values
MAC	Physical address For more information, see section Examples 3599.
SourceMAC	Physical address
DestinationMAC	Physical address
EtherType	IPV4, ARP, RARP, APPLE, AARP, IPV6, IPXold, IPX, any number
VlanPCP	IEEE 802.1Q VLAN Priority Code Point
VlanID	IEEE 802.1Q VLAN Identifier
TrafficClass	IPv6 Traffic Class: corresponds to TOS used with IPv4
FlowLabel	IPv6 Flow Label

Additional Fields for NetFlow v5 and jFlow v5 Sensors Only

Field	Possible Filter Values
Interface	Any number
ASI	Any number
InboundInterface	Any number
OutboundInterface	Any number
SenderIP	IP address of the sending device. Use this if you have several devices that send flow data on the same port, and you want to divide the traffic of each device into a different channel. Possible values: IP address or DNS name For more information, see section Valid Data Formats 3599.
SourceASI	Any number

Field	Possible Filter Values
DestinationASI	Any number

Additional Fields for NetFlow v9 and IPFIX Sensors Only

Field	Possible Filter Values
Interface	Any number
ASI	Any number
InboundInterface	Any number
OutboundInterface	Any number
SenderIP	IP address of the sending device. Use this if you have several devices that send flow data on the same port, and you want to divide the traffic of each device into a different channel. Possible values: IP address or DNS name For more information, see section Valid Data Formats 3599.
SourceASI	Any number
DestinationASI	Any number
MAC	Physical address
SourceMAC	Physical address
DestinationMAC	Physical address
Mask	Mask values represent subnet masks in the form of a single number (number of contiguous bits).
DestinationMask	Mask values represent subnet masks in the form of a single number (number of contiguous bits).
NextHop	IP address or DNS name
VLAN	VLAN values represent a VLAN identifier (any number).

Field	Possible Filter Values
SourceVLAN	VLAN values represent a VLAN identifier (any number).
DestinationVLAN	VLAN values represent a VLAN identifier (any number).

Additional Fields for sFlow Sensors Only

Field	Possible Filter Values
Interface	Any number
InboundInterface	Any number
OutboundInterface	Any number
SenderIP	IP address of the sending device. Use this if you have several devices that send flow data on the same port, and you want to divide the traffic of each device into a different channel. Possible values: IP address or DNS name For more information, see section Valid Data Formats 3599.
MAC	Physical address
SourceMAC	Physical address
DestinationMAC	Physical address

Valid Data Formats

- IP fields support wildcards (*), range (10-20) and hostmask (/10, /255.255.0.0) syntax, as well as DNS names.
  IP fields do not support IPv6 wildcards, IPv6 ranges, and IPv6 hostmasks.
- Number fields support range (80-88) syntax.
- Protocol and EtherType fields support numbers and a list of predefined constants.

For detailed information on IP address ranges, see section Define IP Address Ranges 3603.

Examples

All of the following filter rules are valid examples:

```
SourceIP[10.0.0.1]
SourceIP[10.*.*.*]
SourceIP[10.0.0.0/10]
DestinationIP[10.0.0.120-130]
DestinationPort[80-88]
Protocol[UDP]
MAC[00-60-50-X0-00-01]
DSCP[46]
```

You can create more complex expressions by using parentheses ( ) and the words and, or, or and not. For example, these are valid filter rules:

```
Protocol[TCP] and DestinationIP[10.0.0.1]
```

This rule filters for all TCP traffic with the destination IP address 10.0.0.1.

```
Protocol[TCP] or DestinationIP[10.0.0.1]
```

This rule filters for all TCP traffic and all traffic with the destination IP address 10.0.0.1.

```
Protocol[TCP] and (DestinationIP[10.0.0.1] or SourceIP[10.0.0.120-130])
```

This rule filters for all TCP traffic with either the destination IP address 10.0.0.1 or the source IP address range 10.0.0.120-130.

```
Protocol[TCP] and not (DestinationIP[10.0.0.1] or SourceIP[10.0.0.120-130])
```

This rule filters for all TCP traffic that does not have the destination IP address 10.0.0.1 and the source IP address range 10.0.0.120-130.

More

KNOWLEDGE BASE

How can I change the default groups and channels for flow and Packet Sniffer sensors?

- https://kb.paessler.com/en/topic/60203