

# Scheduling Reports

- [Introduction](#)
- [How to define Report Schedules](#)
  - [In the GUI](#)
  - [Using Schedule Files](#)
- [Scheduler Operation](#)
  - [Recurring Schedules](#)
  - [One-Off Reports](#) ( [One-Off Reports are generated using the Create one-off Report button in opReports 4.3.1 and newer](#) )
- [Remote-Controlling NMIS](#)
- [Optional Settings](#)
- [Report-Specific Settings](#)
- [Output File Naming](#)

## Introduction

Historically opReports has relied solely on being run on demand via the GUI (deprecated in opReports 4.3.1) or command line (or periodically with cron). In version 3.0 we've added another feature that extends these capabilities: the opReports Scheduler. The scheduler is meant to act a bit like a combination of "cron" and "at" (a venerable but not especially well known Unix tool, companion of cron) for opReports.

The basic idea is that you control the scheduler by defining opReports "jobs" to be taken care of (once, at a particular time in the future, or recurring with specific frequencies and start/end times/dates).

## How to define Report Schedules

### In the GUI

As of version 3.0.2 using the GUI schedule editor is the most convenient method for managing report schedules. The GUI schedule editor is a front-end that handles schedule files, so the technical details given in the next section remain applicable.

Version 3.14 [Traffic Snapshot Report](#) can be defined in the Gui. When this report type is selected you can add and delete snapshot pages under (Sources)

### Using Schedule Files

For ease of automated deployment and scriptability opReports provides access to all job schedule options using schedule files.

Each scheduled job is defined by a separate JSON file. The scheduler expects these files in `/usr/local/omk/conf/schedule` (but this is configurable - see `<omk_schedule>` in `conf/opCommon.nmis`).

All job files must be called `<something>.json`, and only letters, digits, dot, underscore and hyphen are allowed in the file name. All other files are silently ignored. You can thus disable a job by naming it something like `!disabled_job.json`. opReports ships with a number of example schedule files (all set to be inactive) and a brief README file in the `conf/schedule` directory.

A job file must consist of one "hash" datastructure, with the following properties:

Property	Example	Description
type	"util" "wan"	The report type to generate. Required. See the <a href="#">opReports Report Descriptions</a> for a list of available types.
active	"true" or "false"	Whether this schedule is active or not. Only active schedules can create new reports, inactive schedules are consulted only for pruning of old report data. If active is not specified, then the schedule is inactive.
description	"some text"	A free-form description of the scheduled job. Optional.
frequency	"daily", "weekly", "monthly" or "yearly"	Not present for one-off reports.
start	"1-14-2015 20:00:00", "mon 10:30", various others depending on the frequency	The date + time the report collection is to start at. Required. See the <a href="#">Formats for Report Periods and Frequencies</a> page for details.
end		The date + time the report collection is to end at. Same format as <code>start</code> .

from	13 or any integer between 0 and 24	Start of Business Hours reporting. Optional. Default: 0. See the <a href="#">Formats for Report Periods and Frequencies</a> page for details.
to		End of Business Hours reporting. Optional. Default: 24.
exclude	"fri-sun", "sat-mon"	Business week selection. Optional, supported in opReports 3.0.8 and newer.
node_intf_list	/some/where/some.file	The file describing the nodes and interfaces to collect. See section below for format.
util_threshold	80	The desired interface utilisation threshold value (in percent), default: 80 (percent).
util_threshold_mincount	1	The desired minimum number of threshold exceedences for flagging the interface as over-limit, default: 1.
format	a hash with the keys "html", "csv" and "xlsx", and values true/false or 1/0. e.g. { "html":0, "csv":true, "xlsx":false }	The formats the report should be saved in.
outputdir	/some/where/	Where the report should be saved. This has to be a full path pointing to a directory. GUI-scheduled reports will always use one of the configured directories (see <code>opreports_output_dirs</code> in <code>opCommon.nmis</code> ), and it is recommended that you use only these even if editing schedule files by hand. Every report schedule can use a different output directory.
naming	"simple" or "precise"	What naming scheme should be used for the output files. Default is "simple". Mainly important if you plan to manually work with saved report files. See the section on Output File Naming below for details.
sources and (at most) one of group_regexp, node_list, node_regexp, node_group, node_intf_list or node_intf_type_list	"everything"	See <a href="#">How to select Nodes (and Interfaces)</a> for details.
keep_for	47	How many days to keep an old report instance. If set to zero, the report is not expired. If not set, then the defaults for the report type are applied (configuration entry <code>default_report_keep_for</code> ).
options	{ "title" : "My Custom ReportTitle" }	Optional settings. Some are specific to particular report types. See the section on Optional Settings below.
control_nmis	"true" or "false"	Whether NMIS should be "remote controlled" or left in peace. See section on Remote Controlling NMIS below. Only relevant for one-off reports.
nmis_options	"mthread=true maxthreads=15"	Options to be given to <code>nmis.pl type=update</code> . Optional, relevant only if <code>control_nmis</code> is true.
target_audience_group	"HQ"	If present and a known NMIS group name, then the generated report can be viewed only by users who are members of this group (and the administrator). If not present then report viewing is not limited to particular groups.

Besides these user-definable properties, the scheduler also manages certain others for internal use, and you should not modify these! (Currently these include the properties "uuid", "in\_progress", "completed" and all properties named "actual\_<something>".)

## Scheduler Operation

When the scheduler starts it looks for jobs that are in need of actions (ie. the wanted reporting period has just passed and the relevant report hasn't been created yet) and handles them sequentially. When the scheduled job is finished, the job file is **not** removed and remains behind (regardless of whether this is a one-off or recurring schedule). After that all known saved reports are checked for pruning: if the report was created with a non-zero `keep_for` property, and is older than this cutoff then it is removed. This affects only report instances, not the job schedules.

The opReports scheduler is not a long-lived daemon process, but rather meant to be run suitably frequently from cron. The scheduler does log to `opReports.log`, but also prints error messages of higher urgency to `STDERR`, where cron will pick them up and mail them to `root` (or whatever recipient you have set up). By default opReports creates an `/etc/cron.d/opreports` which triggers the scheduler once every 5 minutes for recurring reports and every minute for one-off reports.

## Recurring Schedules

Recurring jobs are processed repeatedly, as soon as the scheduler determines that the end of the report period for that schedule is past.

## One-Off Reports ( One-Off Reports are generated using the **Create one-off Report** button in opReports 4.3.1 and newer )

A one-off job is normally processed *exactly once*, as soon as the report period has passed. The job is then marked `completed` and will not be (re) generated.

It is, however, possible to generate another instance of a one-off schedule:

- This can be desirable if your one-off uses relative start and end times (e.g. 'now - 10 days' to 'midnight').
- The scheduler does not attempt to guess that your one-off report *may* be rerun safely; it needs to be told to reconsider the schedule. This is achieved by clearing the `completed` flag in the report schedule.
- If you work on the schedule files directly, simply remove the `completed` property and save the modified file.
- In the GUI, simply open the respective schedule for editing and hit the Save button; this also clears the `completed` property.

## Remote-Controlling NMIS

This *experimental* feature is primarily meant for Interface Utilisation Reports at this time and works only for one-off jobs. If your job has `control_nmis` true AND a `node_intf_list` file, then the following actions are taken:

- When the job starts, NMIS is reconfigured with `global_collect` true and with **ONLY** the wanted nodes and interfaces (identified by the `node_intf_list` file) set to `collect` true. All other interface collection activity is disabled with `global_nocollect_ifDescr`. An `nmis.pl type=update` run will then be performed, and the NMIS data collection commences as normal.
- When the job completes, NMIS is disabled, with `global_collect` set to false.
- This clearly interferes with recurring schedules and should therefore only be used in isolation.

As of version 3.0.2 some limitations apply for this feature:

- The NMIS remote control feature is not supported for recurring report schedules, only for one-off reports.
- The only possible extra action to take at the beginning of a job is enabling NMIS and setting the listed interfaces and nodes for collection.

## Optional Settings

The `options` property can hold sub-properties that fine-tune opReports' behavior. There are a few common options that are shared by all reports:

- Option `title` lets you specify a custom title for the report. The value must be a string.
- Option `homelink` lets you specify a custom banner, to be displayed **only** for a standalone HTML report and above the actual report (title and body).  
The value of this option must be a string that contains the raw HTML to be shown.
- Option `email` lets you specify one or more recipients for report delivery by email. The value must be a comma separated list of email addresses. opReports will email the report to all listed email addresses, with all selected output formats as attachments.
- Option `node_naming` lets you change how node names are displayed.  
If not set (the default), nodes are displayed with their plain node name.  
If set to `display_name`, the NMIS property `display_name` will be used if it's present for a node; if not, the plain node name is used.  
This is supported in opReports 3.0.8 and NMIS 8.6 onward.
- Option `interface_naming` lets you change how interface names are displayed.  
If not set or set to the default value `Description`, the interface's `Description` property is used (in NMIS that corresponds to the `ifDescr` or `ifAlias` property).  
If set to a different interface property name, that property's value is used.  
This is supported in opReports 3.0.10 and NMIS 8.6 onward.

The Node Availability and Interface Capacity reports support option `embedgraphs` (value true or false), which controls whether graphs are embedded into the graph. The [Report Descriptions](#) page describes this feature in more detail.

The WAN report supports option `wan_report_level`, whose value selects which report detail level should be chosen. This can be given either as the index of the `level` or as its name. The page about [WAN report detail levels](#) describes this feature in detail.

The Node Health report has option `exceptions`, which when set to true, changes the report to only display health exceptions and suppress nodes with ok health values.

The Utilisation report recognizes the following four options:

Option	Description
<code>show_threshold</code>	Default: true. If this is false, then no over threshold counts and exceptions are shown. Instead the interface bandwidths, traffic, utilisation and short report period are shown.
<code>show_only_util</code>	Default: false. Ignored if <code>show_threshold</code> isn't false. If this is set to true, then opReports omits the bandwidth and traffic columns and shows only the utilisation (and short period).

util_threshold	Default: 80 (%). Defines the level of utilisation above which it counts as exception.
util_threshold_mincount	Default: 1. How often the interface utilisation has to be above the threshold value for the interface to be flagged as in exceptional state.

The Traffic Snapshot report recognizes the following two *global* options, and numerous [report-specific settings](#).

Option	Description
peak_type	One of combined, busiest_bits or busiest_util. Default is combined. Combined: The peak utilisation for the interface group is defined as the sum of all involved interfaces' traffic. Busiest by Bits: The peak utilisation is sourced from the one interface with the highest traffic figure. Busiest by Utilisation: The peak utilisation is sourced from the one interface with the highest ratio of traffic to configured interface capacity.
capacity_type	One of combined or slowest. Default is combined. Combined: The overall capacity for the interface group is defined as the sum of all involved interfaces' configured capacity. Slowest: The overall capacity is defined as the capacity of the <i>slowest</i> interface, ie. the one with the lowest configured capacity.

## Report-Specific Settings

For Traffic Snapshot reports, please check the [Traffic Snapshot Report](#) detail page.

## Output File Naming

We recommend that you let opReports manage output files completely, and that you make use of the automatic report emailing feature or the Zip button in the GUI to access report instance files.

If that is not feasible for you, the following description of how opReports generates file names may come handy.

Please do note that the behaviour of all naming options except "simple" may change between opReports versions.

1. The file name auto-generation logic only affects the basename of the file; extensions are always set to .json for the report metadata file and .html, .csv and .xlsx for the different output formats.
2. Please note that any graph files associated with a report are **not** subject to this naming scheme; those names are completely dynamic and can be determined only by analysing the report metadata file.
3. On-demand reports generated from the GUI allow the user to supply the 'report name', and if that is present then it'll be used for the file names (but subject to de-clashing as outlined below).  
If no report name is supplied, then On-demand reports are treated like scheduled non-recurring/one-off reports.
4. All file names may be amended with a number if a clashing file already exists, e.g. "some\_report.3.xlsx" and "some\_report.4.xlsx" may be generated.
5. All file names are adjusted to replace certain characters: all spaces and ":" are replaced by "\_".
6. Depending on the report frequency, the following components are used to build the file name:

Frequency	Simple Naming	Precise Naming
any	first the report type, then a "_"	same as simple
any	n/a	if the report schedule selects nodes by group: the group name and a "_"
one-off	the start and end times in ISO8601 format, separated by "_" and followed by a "_"	same as simple
daily	'daily_', then the start and end times in the original format, separated by "-", then a "_", then the end date's abbreviated month name suffixed with the day of the month, a "_", then the end date's four-digit year	same as simple
weekly	the abbreviated start date's month name suffixed with the day of month, a "-", then the same for the end date, a "_", then the four-digit year	'weekly_', the start and end date/times in the original format separated by hyphen (e.g. Tue_00-Fri_14_45 after space removal), a "_", then the end date's abbreviated month suffixed with the day of the month, then a "_" and finally the end date's four-digit year

monthly	the abbreviated month name of the end date, a "_", then the four-digit year	'monthly_', the start and end dates/times in the original format separated by hyphen (e.g. 4_17_00-30_24_00 after space removal), a "_", then the end date's abbreviated month name suffixed with the day of the month, another "_" and finally the end date's four-digit year.
yearly	'yearly_', the start and end dates in the original format separated by hyphen (e.g. "12_01_14_00-12_24_19_00" after space translation), a "_", then the end date's abbreviated month name suffixed with the day of the month, another "_" and finally the end date's four-digit year.	same as simple