Purging of old data in opConfig

Introduction

Versions 2.2 (and newer) of opConfig provide more detailed collection and display of operational status, and now also let you control whether and when old data should be expired. This document describes how to configure these features.

Operational Status

opConfig's new status display (Views -> Operational Status) is a complement to the plain text logs, to make it easier to get an overview of what opConfig is doing to which nodes, when, and how successful it was with these operations. This material grows very quickly and expiration is a must.

To configure expiration you need to set the maximum age of status entries is set in `conf/opCommon.nmis`:

```
'opconfig' => {
...
  'opconfig_opstatus_maxage' => 604800, # seconds, 1 week
```

You may want to reduce this value if your opConfig deployment manages lots of machines. The status material is pruned automatically whenever opConfig performs any kind of operation (GUI or opconfig-cli), if the config entry is set.

Command Revisions

By default opConfig does not remove old revisions from the database; if you run many non-change-detecting commands (or encounter frequent changes in your commands' outputs) then this will likely make the GUI unwieldy (e.g. the revision drop down menus will become very large).

opConfig versions 2.2 and newer let you control the purging of old data in a very flexible fashion.

Protecting Revisions

By default all revisions are unprotected and subject to purging.

Using the GUI

To ensure that a particular revision of a command for a node is not expired and removed, you can set its status to "protected" using the GUI.

To do so, navigate to the Command Output page (click on a command or change entry on one of the overviews, or select Views -> Command Output). The Command Summary panel on the right shows the revision in question, and a Protected/Unprotected status button besides it. Simply click the button to toggle the protection status.

Automatic Protection

If a command in your command sets has (or has inherited) the property `autoprotect_first_revision` with value true or 1, then the first revision of a command (for a node) will be marked as "protected". All subsequent revisions will be unprotected. This is useful for establishing a baseline status that you don't want to be removed regardless of its age.

The automatic protection setting establishes only a default protection status; You can still change the status later in GUI.

(Less useful but mentioned for completeness' sake: If you add the tag "`_protected`" to a command's tags in your command sets, then every new revision for this command will start in the "protected" state.)

The `autoprotect_first_revision` flag can be set

1. globally, as `opconfig_purging_autoprotect_first_revision` in `conf/opCommon.nmis`,
2. or for one command set, in the `purging_policy` section for that set in `conf/command_sets.nmis`,
3. or for a particular command, as property `autoprotect_first_revision` within this command's definition section in `conf/command_sets.nmis`.

The settings are inherited, and more specific settings override the defaults.

Please note that this feature **does not work retroactively** and does not adjust the protection status of revisions that were captured in the past.

Purging Policy

There are two criteria for purging of old data:

- the number of revisions to keep, controlled by the property `keep_last`,
- and the maximum age of a revision, controlled by the property `purge_older_than`.

To explicitly disable `keep_last` or `purge_older_than` simply set the value to 0.

You can use these independently or combined. If both are active, then a revision must meet both criteria before being expired.

For example, setting `keep_last` = 100 and `purge_older_than` = 2592000 (1 month, in seconds) would keep at least the most recent 100 revisions and at least the last month's data - or, in other words, it would prune only revisions that are *both* outside the newest 100 *and* older than a month.

Protected revisions are not considered or counted and left untouched. The age of a revision is based on its "last updated" timestamp.

When a revision is removed, it also remove the associated job, if no other revision are linked to that job, since opConfig 3.5.1.

Inheritance, Setting up defaults

There are three levels of purging policy:

1. Global
2. Command Set
3. Command

Global

You can set a global default purging policy in `conf/opCommon.nmis`:

```
'opconfig' => {
 ...
 'opconfig_purging_keep_last' => 100,        # keep 100 most recent revisions
 'opconfig_purging_purge_older_than' => 31536000, # purge revisions older than 1 year
 'opconfig_purging_autoprotect_first_revision' => 'true', # protect the first revision by default
```

Command Set

You can extend or overrule that default for a whole command set, in `conf/command_sets.d/{vendor-name}.nmis`:

```
'IOS_HOURLY' => {                  # command set name
  ...
  'purging_policy' => {
    'keep_last' => 1000,
    'purge_older_than' => 2592000, # 30 days
    'autoprotect_first_revision' => 'true',
  },
```

This example overrules all the global defaults for all commands in the IOS_HOURLY set, with a shorter retaining period (but a higher number of minimum entries).

Command

Finally, you can also set a purging policy for a single command in a command set only, again overriding any of the command set or the global defaults. Here is such an example, again configured in `conf/command_sets.d/{vendor-name}.nmis`:

```
'TS_LINUX_PROCESSES' => {         # command set name
  ...
  'commands' => [
    {
      'multipage' => 'true',
      'privileged' => 'true',
      'command' => 'ps -ef',      # command
          'keep_last' => 0,
          'purge_older_than' => 86400,
      'tags' => [ 'troubleshooting', 'operatingsystem' ],
    },
  ],
},
```

This example makes sure that for the command `"ps -ef"` in the TS_LINUX_PROCESSES command set, only the last day's revisions are kept, regardless of any global policy or policy for the TS_LINUX_PROCESSES set. Note that `keep_last` needs to be disabled explicitly here, or the defaults for this setting would be inherited.

Performing the purging

The purging operation is **not automatic** and needs to be triggered with `opconfig-cli.pl`, ideally from a cron job:

```
# /etc/cron.d/opconfig for example
40 3 * * *     root /usr/local/omk/bin/opconfig-cli-pl act=purge_revisions quiet=1
```

This cron entry would run a daily purge at 03:40.

Jobs

The jobs are purged every time a revision is purged, if there are no other active revisions linked to the job. This was modified since version 3.5.1.

But there is another cli tool, to remove jobs with no revisions associated. This jobs are purged based on the configuration option:

`opconfig_queue_expire_after_seconds`

Set to 8 days by default => 86400 * 8. The purging operation is **not automatic** and needs to be triggered with `opconfig-cli.pl`, ideally from a cron job:

```
# /etc/cron.d/opconfig for example
40 3 * * *     root /usr/local/omk/bin/opconfig-cli-pl act=purge_queue
```