Node Administration with opnode_admin

NOTE: This documentation is only relevant to Opmantek Products with NMIS8, when using NMIS9 node_admin.pl performs all necessary node administration functions for NMIS, opConfig and opEvents.

For NMIS9 and op Modules

- NMIS Node Administration Tools

Table of Contents

opConfig and opEvents provide a number of different methods for managing nodes, both GUI-based and commandline-oriented. This document describes the commandline tool `opnode_admin.pl`.

Basic Operation

The `opnode_admin` tool is installed by default as `/usr/local/omk/bin/opnode_admin.pl`. It is scriptable and pipelineable, and can perform all node-related operations: creation, updating, renaming, exporting and deletion of nodes.

Run the tool with no options or -? or -h and it'll display a simple help page:

```
/usr/local/nmis9/admin/opnode_admin.pl
Usage: opnode_admin.pl act=[action to take] [extras...]
        opnode_admin.pl act=list
        opnode_admin.pl act={create|export|update} node=nodeX
        opnode_admin.pl act=delete node=nodeX [deletedata=0/1]
        opnode_admin.pl act=show node=nodeX
        opnode_admin.pl act=set node=nodeX entry.propname=value...
        opnode_admin.pl act=mktemplate [placeholder=1/0]
        opnode_admin.pl act=rename old=nodeX new=nodeY
mktemplate: prints blank template for node creation,
 optionally with __REPLACE_XX__ placeholder
create: requires file=NewNodeDef.json
export: exports to file=someFile.json (or STDOUT if no file given)
update: updates existing node from file=someFile.json (or STDIN)
delete: only deletes if confirm=yes (in uppercase) is given
node events and config information are deleted only if deletedata
is 1 or true.
show: prints the nodes properties in the same format as set
set: adjust one or more node properties
```

Node Listing, Viewing and Exporting

Given the argument `act=list`, `opnode_admin` will print a list of all known node names.

Version 2.70.0 onwards prints its two lines of header only when displaying to a terminal; with older versions you'll have to exclude these lines if you want to reuse the data.

To save a node's config information, run `admin/opnode_admin.pl act=export node=`*`somenodename`*, and it'll print the node's configuration in JSON format to your console. If you want to save that data in a file, either add the argument `file=`*`somefilename`*`.json`. Here is an example of what to expect:

```
./bin/opnode_admin.pl act=list
Node Names:
===========
bne-server1
bne-switch1
build
char-router1
...
```

To see a node's properties in a relatively human-friendly form you should use `act=show`; it'll present the properties in a 'dotted-path' notation, like this:

```
./bin/opnode_admin.pl act=show node=mel-router1
entry.active=true
entry.activated.opConfig=1
entry.activated.opEvents=1
entry.customer=Sales
entry.group=Melbourne
entry.host=mel-router1
entry.nodeModel=CiscoRouter
entry.nodeType=router
entry.nodeVendor=Cisco Systems
entry.nodedown=false
entry.processorRam=99.61 Mb
entry.softwareImage=C1841-ADVENTERPRISEK9-M
entry.softwareVersion=12.4(25f)
entry.sysContact=default
...
```

Exporting all of a node's configuration (in JSON format) works like this, if you don't specify an output file to save the data in:

```
./bin/opnode_admin.pl act=export node=mel-router1
{
   "InstalledModems" : 0,
   "active" : "true",
   "addresses" : [],
   "customer" : "Sales",
   "depend" : "N/A",
   "group" : "Melbourne",
   "host" : "mel-router1",
   "host_addr" : "",
...
```

Node Updating

Naturally `opnode_admin` does not just export node data, but also consumes it for modifying a node in place and for creation of new nodes. There are two modification operations: `act=set` and `act=update`:

- `act=set` works best for adjusting a few individual properties.
  Its advantage is that doesn't require a full export of the node data.
- `act=update` works best for big modifications, and expects you to supply a complete node config in JSON format.

Relative updates, changing just a few properties

For example, to change a node's opConfig activation field to false (so that opConfig does not attempt to communicate with the node) you'd certainly use `act=set`, like this:

```
./bin/opnode_admin.pl act=set node=mel-router1 entry.activated.opConfig=0
Successfully updated node "mel-router1"
```

Any number of arguments of the form `entry.<propertypath>=<somevalue>` can be given, and all identified properties will be updated in one operation.

Please note that you cannot set authentication-related sensitive properties (like the snmp community, for example) using `opnode_admin act=set`; these are ignored and a warning is shown.

Absolute updates, setting all properties

To change all of a node's configuration (**except** node renaming!), simply dump the node configuration with `act=export`, then edit the exported data suitably and finally perform the change with `act=update` and the exported data as input. Both operation require that you give the node name in question, and both work either from files (with a `file=`*`somefile`*`.json` argument), or via STDOUT/STDIN/pipeline.

For example, this pipelined invocation would change a node's misspelled group property:

```
./bin/opnode_admin.pl act=export node=mel-router1 | sed -e 's/WorngGroup/correctgroupvalue/' | ./bin
/opnode_admin.pl act=update node=mel-router1
```

You can also explicitely pass the argument `file=-` to indicate that STDOUT should be used for export or STDIN be used for update/creation. Please note that the `act=update` operation doesn't create new nodes or rename noes, and that it **replaces** the whole set of node configuration settings with your new configuration input.

Creation of Nodes

Node creation is triggered by the argument `act=create`, which behaves mostly like `act=update`, except that it doesn't touch existing nodes. To help you with starting a node configuration document from scratch (or in a scripted fashion), there is another command, `act=mktemplate`, which prints a blank but documented template which you can save and fill in. If you add `placeholder=1` to the command line, then `opnode_admin` fills the template with easily matchable replacement placeholders, like so:

```
./bin/opnode_admin.pl act=mktemplate placeholder=true
// Please see https://community.opmantek.com/display/opCommon/Common+Node+Properties for detailed descriptions
of the properties!
...lots more hopefully helpful comments
{
   "activated" : {
      "opAddress" : "__REPLACE_opAddress_ACTIVATION__",
...
```

This makes it very easy to fill in the template with a script or some other external tool.

Node Renaming

To rename nodes you should use `act=rename` which requires both old and new node names with arguments `old` and `new`, respectively. This operation first changes the configured node name, and then adjusts most database entries related to the node in question:

- If opEvents is installed and licensed, all existing events for the renamed node will be rewritten to refer to the new node name.
- if opConfig is installed and licensed, then all existing command outputs for the renamed node will be updated to refer to the new node name.

Deleting Nodes

To remove a node (but not its historic data) simply run `opnode_admin` with the argument `act=delete node=`*`ripnode`*, plus the option `confirm=YES` (must be uppercase) to make opnode_admin actually perform the deletion.

This removes only the node configuration information but not existing opEvents or opConfig data for the node.To delete these as well, you can add the option `deletedata=1` to the command, and all data related to this node will be removed permanently.