NMIS 9 Collect and Update Plugins

- collect_plugin and update_plugin
- after_update_plugin() and after_collect_plugin()
- How to access the information in NMIS9
  - The following all assumes the following function initialization
  - Nodes
  - Collected Data - As Interfaces
  - Log
- Installation
- Example
- Porting Plugins from NMIS8 to NMIS9

NMIS code plugins are meant to augment and extend the update and collect functionality, if and when classic modelling doesn't suffice to handle a particular scenario.

Each plugin must be valid perl, be named X.pm, must have a 'package X;' line, and its package name must not clash with the other NMIS components. the plugin MUST NOT use Exporter to export anything from its namespace.

It's recommended that the plugin have a version declaration right after the package line, i.e. 'our $VERSION = "1.2.3";'

The plugin may load extra perl modules with use or require, but it must not use any package-level global variables. all its variables and any objects that it might create must have local scope.

A plugin can offer one or more of the functions update_plugin(), collect_plugin() after_update_plugin() or after_collect_plugin().

They work pretty much the same as in NMIS 8, but the API calls change.

collect_plugin and update_plugin

These functions will be called for each node in sequence, independently (and possibly in separate processes), at the end of the collect/update operation. all existing valid plugins will be loaded, and all available update_plugin()/collect_plugin() functions will be applied to every node regardless of the plugins success or failure indications.

The calling interface is the same for both functions:

- sub update_plugin(node => nodename, sys => live sys object, config => read-only configuration object/hash, nmisng => nmis object to obtain node data)
- sub collect_plugin(node => nodename, sys => live sys object, config => configuration object/hash, treat as read-only, nmisng => nmis object to obtain node data)

The plugin can modify any node attributes via sys where required. it may call other nmis code (but care needs to be taken of nmmis config/table caching if that is done). neither the local config object nor the global nmis config must be modified by a plugin.

return values:

- (0 or undef,rest ignored) means no changes were made, or that the plugin declined to run altogether (because this node has a non-matching model for example)
- (1,rest ignored) means changes were made to the sys object, and nmis will save the nodeinfo and view components.
- (2,list of error messages) means the plugin failed to work and NMIS should please log the error messages. no nodeinfo/view saving will be performed.

after_update_plugin() and after_collect_plugin()

These functions will be called ONCE at the end of the respective operation, after all the nodes were handled. there is thus no node context for these plugin functions.

Just like with the per-node functions, all existing valid plugins will be loaded and all available after_update_plugin()/after_collect_plugin() functions will be run in sequence.

The calling interface is the same for both functions:

- sub after_update_plugin(nodes => list of node names that were handled, nodes_uuids => list of node uuids, sys => live sys object, config => read-only configuration object/hash, nmisng => nmis object to obtain node data)
- sub after_collect_plugin(nodes => list of node names that were handled, nodes_uuids => list of node uuids, sys => live sys object, config => configuration object/hash, treat as read-only, nmisng => nmis object to obtain node data)

The sys object is the 'global' one used for calculating system-wide metrics and so on. the plugin can modify any attributes via sys where required. it may call other nmis code (but care needs to be taken of nmis config/table caching if that is done). neither the local config object nor the global nmis config must be modified by a plugin.

return values:

- (0 or undef,rest ignored) means no changes were made, or that the plugin declined to run altogether (for example, because the environment doesn't match the plugin's needs
- (1,rest ignored) means changes were made to the sys object, and nmis should save the nodeinfo and view components.
- (2,list of error messages) means the plugin failed to work and NMIS should please log the error messages. no nodeinfo/view saving will be performed.

How to access the information in NMIS9

The following all assumes the following function initialization

```
my (%args) = @_;
my ($node, $S, $C, $NG) = @args{qw(node sys config nmisng)};
```

Nodes

We can access the node object using the object sys:

```
my $node_obj = $S->nmisng_node
```

Collected Data - As Interfaces

We All the collected data is saved in a new structure called inventory. We can use as many filters that we need:

```
my $host_ids = $node_obj->get_inventory_ids(
                concept => "Host_Storage",
                filter => { historic => 0 });
```

Log

We can log any message by using the NMISNG object:

```
$NG->log->info("Working on $node Host Memory Calculations");
```

Installation

Just place the plugin into /usr/local/nmis9/conf/plugins and make sure it has the right permissions.

To fix permissions you can just run:

```
/usr/local/nmis9/bin/nmis-cli act=fixperms
```

You can test the plugin by running an update/collect for a node and check the log files (/usr/local/nmis9/logs/nmis.log).

You will see an info message like this:

```
[Mon May 24 16:12:09 2020] [info] Host_Resources[8027] Running Host Resources plugin for node::test
```

Example

You can see a complete example of a collect plugin in the Host_Resources plugin (Released in 9.1.2). There are many more plugins added to the NMIS9 project.

Porting Plugins from NMIS8 to NMIS9

Existing plugins in NMIS8 were ported to NMIS9, so there are many good examples on the changes to the API calls required. You can find all the ported and any new plugins in GitHub here: https://github.com/Opmantek/nmis9/tree/nmis9_dev/conf-default/plugins

Review the NMIS8 Host_Resources and NMIS9 Host_Resources plugin as a way to see how to easily port one.

If you have a support contract and need some assistance to port your plugin, you can open a support case: FirstWave Support