NMIS Collect and Update Plugins for Flexible Data Collection

The following is included the README file of the NMIS8 source code, you can find the original here: https://github.com/Opmantek/nmis8/blob/nmis8_dev/install/plugins/README

For a NMIS 9 reference guide, please check the following documentation.

NMIS plugins

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().

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)

sub collect_plugin(node => nodename, sys => live sys object,
config => configuration object/hash, treat as read-only)
```

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 nodes that were handled,
sys => live sys object,
config => read-only configuration object/hash)

sub after_collect_plugin(nodes => list of nodes that were handled,
sys => live sys object,
config => configuration object/hash, treat as read-only)

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.