NetYCE 7.1.1 Build_20190924

Release notes

Date: 2019-09-24

Enhancement

Custom reports scheduling

Custom reports can be scheduled to be executed periodically. The scheduling allowed the time-of-day, date and day-of-the-week to be specified. Options for multiple times per day, dates or weekdays were not available.

Since the report scheduler deploys the standard unix 'crontab' functions, these advanced scheduling functions could be made available for the reports too. The crontab options are powerful but require some familiarity.

The Custom reports tool now includes a text box with the resulting crontab time specification after the (basic) choices are made. This text box can then be modified to include the advanced options as desired. This information is then used to schedule the report. Incorrect settings are rejected.

The crontab format uses five fields:

```
 .---------------- minute (0 - 59)
 |  .------------- hour (0 - 23)
 |  |  .---------- day of month (1 - 31)
 |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
 |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
 |  |  |  |  |
 *  *  *  *  *

Examples:
 00 12 * * *                  = daily at noon
 05 04 * * mon                = mondays at 04:05 in the morning
```

The advanced options allow values to be divided, enumerated and ranged:

```
Examples:
 */10 * * * *                   = every 10 minutes
 00 12 * * mon,tue,wed,thu,fri  = noon on weekdays
```

```
*/15 7-18 * * mon-fri          = every quarter hour during office hours
```

Ansible support

Scenario's now have the ability to support a (local) **Ansible** installation.

The new “Ansible_exec” command for scenarios allow the launching of either an existing Playbook, or the launching of a NetYCE Scenariobook by Ansible. This effective extends the use of NetYCE with the vendors supported by Ansible, while at the same time extending Ansible with the scenarios, parameterisation and modelling of NetYCE.

Where the Playbook option refers to a pre-existing playbook-file that can be used as-is, the Scenariobook refers to the NetYCE template-based generated Playbook that can be parameterised using the NetYCE modelling and database.

The NetYCE GUI supporting these Scenariobooks will soon be extended with dedicated XML and YAML template types.

Scenario 'signal_json'

Scenarios have long had the ability to signal external systems or users using SNMP traps and emails. Now the ability to signal an event using a RESTFUL JSON transaction has been added.

The scenario command “Signal_json” is used to issue a JSON transaction to targets of choice. Details for call (like the authentication to be used) is defined in a dedicated configuration file.

See Scenario commands - Signal_json for details on the use and configuration of this new command.

Context-template syntax

Context templates are used to include sub-templates indirectly based on values in a relation. The relation has for instance the names of the port-templates of a device. Then the context-template will be substituted with the parsed port-templates for all interfaces in the relation.

Common line for a context-template is:

```
-- include loopback interfaces
{<Port_template@Loopbacks_node>, Loopbacks_node}
-- include physical interfaces
{<Port_template@Interfaces>, Interfaces}
```

The syntax for these context-templates has been extended to support multiple conditions. These conditions can be used to filter the appropriate rows of a relation before including the remaining templates.

These conditions use the syntax:

```
{tpl_reference, relation, column=value, column=value, ...}

example:
{<Port_template@Port_map>, Port_map, Port_class='Lo'}
{<Port_template@Port_map>, Port_map, Port_class='Gi', Port_id != '*2*'}
```

All conditions need to be 'true' to be included in the output. Conditions can use the operands '=' and '!=' and the values support wildcards. Conditions using non-existing column names are ignored. There is no limit on the number of conditions.

Scenario 'send-email'

At customer suggestion the 'send_email' command for scenarios has been extended with options for setting the To: and From: email addresses of the issued email.

The implementation allows for multiple To: addresses. By default are the user mail-address and optionally (by tweak) the user-group mail-addresses used for the To: field. from the operator

Scenario 'script_exec'

For scenario's a new command has been added that allows scenarios to launch scripts. This facilitates the use of network automation scripts that engineers build to execute specific tasks. By integrating these scripts, the effort that originally went into the development can be reused.

The 'script_exec' command will execute any file one the NetYCE server for which the interpreter is installed or is a compiled binary. These interpreters include python, perl, bash and sh by default, but can be extended. The python support is version 2.6 for CentOS 6.x and version 2.7 and 3.x for CentOS 7.x.

The 'script_exec' command includes an option for overriding the interpreter and a timeout to forcibly kill the script after an interval. Script arguments can be passed in using multiple options.

Scenario 'grep'

When creating scenarios involving list variables, the 'grep' function is a useful command to extract desired items. This function was reworked to accept filter strings in two distinct formats: wildcards ('*' and '?') and regular expressions (regex).

The wildcard format is widely used and will satisfy most demands. Wildcard filters are case-insensitive:

```
# filter the entries using WILDCARDS in the -c argument
<nie_names> := grep -l <all_names> -c "m*nie"
```

For more advanced filters the more complex and versatile regex format is available. To distinguish the condition argument must be placed between slashes ('/.../') and accepts modifiers like 'i' to compare case-insensitive:

```
# filter the entries using REGEX
<nie_names> := grep -l <all_names> -c "/m.*nie$/i"
```

For details and examples, please consult the wiki page on Scenario Command details - grep

Scenario 'match'

In addition to the 'grep' function, the 'match' function was added to the scenarios.

The behaviour of 'match' is similar to the 'grep' function but with an essential difference: where the 'grep' returns the actual element of the matching condition, the 'match' returns only the matching string. In essence, the 'match' is a grep function and a substring function rolled into one.

When using the regex condition a distinct feature can be used: multiple substring matches. Regular expressions have the capability to 'mark' matched strings using '()' and refer to them by position ($1, $2 and so on). The 'match' function allows you to use this feature to use these marks in the condition regex and have these marked strings returned as a separate entry in the result.

For details and examples, please consult the wiki page on Scenario Command details - match

Supernets for Sites

Automating network designs including IP-address allocations in NetYCE requires the use of IP-plans. IP-plans (ipv4 and ipv6) provide an abstraction for segmenting and standardising ip-spaces that are to be used in the design. They function as a 'blueprint' for allocating required subnets needed to 'build' the various components of the design.

For the implementation an ip-space (or ip-block) in the form of a Supernet is assigned to a NetYCE Client along with the IP-plan that will be governing it. Many large or small IP-plans can be made available for the networking design of the Client.

Because not all designs allow for a 'Client' based approach, customers have requested an option to manage IP-plans and Supernets per Site. Starting this release, Client Supernets can now be 'reserved' to be exclusively used by a Site. The Site will from then on be 'owning' the ip-space of that Supernet.

The assignment to a Site can be done using the GUI by editing a new or existing Supernet in the Client details form. The Site reservation is made by entering a valid SiteCode within the Client. Alternatively the Service-type 'ASSIGN-SUPERNET-SITE_CODE' can be used to further automate this task.

When a subnet for a requested Net_name is to be added, the Net_name is initially searched for in all Supernets 'owned' by the Site. When available in multiple Supernets (regardless of ip-plan), the new subnet is allocated from the Supernet that has the least free subnets of that type (thus filling up used supernets before newer). Should the subnet not be available in 'own' Supernets, the search will continue in Supernets that are not 'owned' by other Sites but are 'shared' within the Client. The same logic in selecting the Supernet then applies when multiple provide the Net_name.

For existing designs set up using Client-'shared' Supernets nothing will change, the

implementation of the Site-supernets is fuly downwards compatible.

Infoblox DNS IPv6

For customers deploying the Infoblox integration another step in integrating IPv6 was implemented.

The module for Infoblox DNS management now allows fo the creation of 'host' records with IPv6 addresses using the NetYCE 'add_host' API call and the scenario function 'dns_add_host'.

This initial implementation is limited 'host' records and the use of a specified IPV6 address. Support for AAAA-records or searching for a 'free' IPv6 address is planned for a subsequent phase.

See the NetYCE Wiki for details in the pages Scenario commands - dns_add_host and Infoblox API plugin - add_host

Menu defaults

The NetYCE front-end menus 'Design', 'Build', 'Operate' and 'Admin' each have a list of forms and tools for the end-user to choose from. It is now possible to configure the system to define which of these items wil be automatically selected when its menu is opened.

The selection what form or tool(set) to open for each menu is controlled using the Lookup “Tweaks” category. The Variables “Default_menu_design”, “Default_menu_build” and so on can be set to specify the item. The values to be used are listed in the Wiki article: Default-menu tweaks.

These Default-menu tweaks can also be used in combination with the Default-tool tweaks that specify which tool from a tool set is selected by default.

Change

Scenario 'config_save'

The scenario command 'config_save' will push the devices' startup-configuration to the NetYCE server. This worked properly but uploaded the existing startup file.

The 'config-save' procedure has been changed such that it will now first create a new startup file and then upload it to the NetYCE server.

Domain permissions

The required privileges to view and edit the Domain form have been modified. Users with Engineer

or Modeller permissions were denied access to the various 'password' type fields. These levels have been loosened.

The following privileges now apply by default:

- Operators can view the non-password fields but all password fields are starred out and disabled.
- Engineers can edit the non-password fields but all password fields are read-only. The password fields appear starred out until it clicked, then the value is displayed.
- Modellers and Managers have full access to all password and non-password fields. Again the passwords

appear starred out until the field becomes active, then it can be edited.

This default behaviour can be modified using the Auth_permissions table where for each of the fields and for all user levels the edit permissions can be modified using the “Admin - Custom data” tool for the “Auth_permissions” table.

```
By filtering on "Controller: 'domains'" and "Type: 'field'", the value
for the desired role can be modified in one of 3, 4, or 5:
  '3' denotes 'disabled',
  '4' results in 'read-only'
  '5' in full access.
```

Help menu items

The 'Help' menu item of all four menus has been removed. The link to the NetYCE Wiki is moved to the upper right hand corner of the page.

This change was made to reduce the number of items in the various menus.

Longer Net_names

At customer request, the 'Net_name' of ipv4 and ipv6 subnets can now be 50 characters long. Previously the maximum length was 20 characters.

Scenario arguments

Scenario functions use arguments like '-a' and '-p' and so on. Some of these functions allow for many (repeated) arguments resulting in very long function calls. Although this offers no functional issues, these long lines of can arguments become unwieldy and look untidy.

Therefore the scenario arguments can now be split over several lines. All lines starting with something that looks like an argument flag (like '-a ' or '-p ') will be taken as an argument belonging to the previous function. This allows for a better overview of the arguments and makes editing a lot easier.

Be aware that the list of arguments cannot contain comments (starting with '#').

Relation test

After the completion of a NetYCE software update or the updating of a system configuration many configuration files are regenerated and processes requiring them are restarted. Because these software updates also install patches on the NetYCE database, the database could be changed in a fashion that potentially renders customer-created Relations non-functional.

Therefore, after all changes were mede and processes are restarted, all existing Relations are checked for errors and the results listed in the Task-log. This process of error-checking of Relations has been modified in two ways:

First, the relation will now use relevant data fields for its substitutions (where possible). Where previously all variable substitutions used '0', it now uses the values from one of the nodes in the database. This results in more relevant queries and catches errors that otherwise would be overlooked.

Secondly, since the queries now run on the database using actual values, the queries will likely respond in a more representative fashion. For all relations the required execution time of the query is measures and reported. For those queries where this exceeds one second, the relation is marked as “Slow” and should be examined for optimisation.

Operator permissions

Users with Operator-level permissions will find they have now read-only access to Templates where they previously could not access this sesion of the menu.

The same change was made to the Mpls-vrfs forms.

Fix

Scenario 'db_query'

The scenario command 'db_query' did not return any resulting values from the requested database table. It turned out that the result was returned in an incorrect format. The db_query command now returns a single value or list of values (array) correctly.

Telnet Vendor modules

Several vendor modules exhibited some shortcomings when using Telnet instead of SSH. If a

username or password failed, the job did not report the origin of the login failure but just aborted.

The issue has been resolved for most vendors, but testing could not be completed for all vendors at this time. This testing is in progress.

HP VPN warning messages

The HP C5 and C7 devices will issue error messages about already existing VPN's. These messages were incorrectly ignored by NetYCE jobs.

These VPN messags are now added to these modules, and depending on the type, marked as an error (stopping additional commands) or a warning (logging it, but continuing).

Export download

Under some circumstances the download of a model export (Service-type, Node-type, Node-groups, Region, Domain, Relations, Scenarios) would result in an empty file. The use of vpn tunnels or firewall settings could cause this.

The issue was caused by the disallowed http data-type 'xml'. By changing it to the the 'octet-stream' data-type, the problem was resolved.

Huawei command parse

When using command parsing to extract information from a Huawei device, the command failed because NetYCE incorrectly enabled the privileged mode where show commands are not allowed.

By modifying the Huawei State_actions table entries for this command, the issue was resolved.

Template conditions

Over the years the supported template syntax has been extended with many features. These enhancements also affected the capabilities for template conditionals. As a consequence some valid conditional syntaxes of the past suddenly became ambiguous or invalid.

To prevent these situations, the NetYCE configuration generator would automatically detect these older style conditionals and convert them on the fly. The kind of corrections are mostly like these:

```
original
|variable|
|variable = value|
|!variable@relation = value1,value2|

corrected
|<variable>|
|<variable> = 'value'|
```

```
|<variable@relation> != ('value1','value2')|
```

Recently the syntax again allowed for more advanced options but were found working fine for data substitution situations (template text), but produced unpredictable results in conditionals. These were found to be the result of the implicit conditional corrections outlined above. The existing conversion was not suited to deal with these syntaxes and would break an otherwise valid conditional.

A complete new conditional converter was created that will still provide the required conversions for the old-style formats while also recognising the sometimes very complicated syntaxes like:

```
|<variable@relation:column=<variable2>> != (<variable3>,'value')|
|[function(<variable1@relation1:'value1')] = <variable2@relation2>|
```

> Update: some valid conditionals in existing templates were incorrectly malformed by the converter.

These cases were identified and fixed. After extensive retesing the resulting converter should now be fully downward compatible.

Scenario variables

Variables defined in the [parameter] section of a scenario are shared with the configuration generator and, though it, the relations. These variables are made available to the generator using an intermediate format.

A malformed variable caused the generator to abort when reading the intermediate format resulting in empty configurations. Extending the validation of external variables was added to resolve this issue.

Scenario 'dns_clear_host'

A problem was found with the 'dns_clear_host' call for Infoblox DNS manipulations. It was found that some hosts addresses could not be found to be removed while the Infoblox front-end clearly showed it existed in the IPAM views.

The issue could be resolved when it became understood that the ip-address was not a regular IPAM address, but consisted of an Infoblox constructed object “FixedAddr” that is used for DHCP address-to-attribute associations like MAC-addresses.

The scenario 'dns_clear_host' and API 'clear_host“ function now locate and remove the “FixedAddr” like a regular ip-address.