NetYCE 7.1.1 Build_20200115

Release notes

Date: 2020-01-15

> This **7.1.1** build is based on the previous 'General release' 7.1.1 build number **20191218**. The only differences are the Hotfix below and a minor adjustment of Zip-code sizes.

Hotfix

Node-groups form

The Node-groups form exhibits a bug that redirects the user to the login page when he selects a Node-group to view or edit.

When we fixed a bug in the Node groups form, the handling of the Node-group Rule attributes was moved to a different module and thereby was made dependent on permissions.

Because these changes were build on the 7.2 code, which will include a caching mechanism for the Node-group nodes, this back-end dependency was overlooked when the front-end fix was ported to the 7.1.1 code base.

To fix, patch 19102101 was moved to the 7.1.1 base and the missing module routine was added.

Enhancement

Device model and s/w version

Although NetYCE allows to model a device software-version and hardware type, it did not track the actually running software-version and hardware-model of each device. To support creating templates and compliance rules for specific hardware or s/w versions these values are now dynamically determined at each job start and stored in the NetYCE database. This is done for both YCE and CMDB based nodes.

These new variables are named <Node_model> and <Software_version>.

Scenario 'Split' function

The 'split' function was added to the scenarios. This function allows a string or variable to be split

into a variable array using a separator character or string. The function accepts multiple input variables or lists of which each element will be split using the specified separator. A single list variable will be returned.

The separator may be a single character, a string using wildcards, or a regex pattern. Optional arguments allow the result to be sorted alphabetically or to return only a limited number of elements.

Scheduler node-locks

The NetYCE job scheduler always accepted jobs for any node and executed the jobs at their assigned time-slot. When two jobs are scheduled for the same node and their execution overlapped, the resulting configuration could become unpredictable and might require longer execution times.

To alleviate these complications, node job-locking is added to the scheduler. This function will prevent the scheduler to execute more than one job at the time for the same node. Jobs slotted to start executing while a NetYCE jobs is still running for that node will be put on hold until the previous jobs finishes. Jobs on hold will have a scheduler state 'wait' which is highlighted using a yellow background.

The node for which the lock is checked is the node selected when submitting the job. The locks do not apply to other nodes the job may open sessions with.

To prevent locking out subsequent jobs because of a very slow job, the maximum duration a node may obtain an exclusive lock is 10 minutes. After this time a waiting job may start regardless of the still running job. This second job may also keep the lock for only 10 minutes.

Should a series of jobs all wait for its predecessors to finish, they can only do so for one hour. After one hour in the 'wait' state, the job is set 'suspended' in the assumption that its maintenance slot is now expired. Suspended jobs can be rescheduled by the operator.

Finally, if the new job-locking by node is undesired, it can be disabled by setting the Lookup Tweak 'Sched_ignore_locks'. The job activation mechanism will then function as in previous releases.

SSL hardening

The NetYCE front-end can be set-up to use the unencrypted 'http' or the encrypted 'https'. When using https, a SSL-certificate must be created and installed. To further strengthen the security offered by https and the certificate, the protocol and encryption ciphers can be chosen to offer best protection.

By default NetYCE accepts TLS1, TLS1.1 and TLS1.2. An additional setting in 'yce_setup' allows the administrator to restrict these protocols to TLS1.2 only. The lower levels are considered weak and the preferred level is TLS1.2 which all modern Browsers support. Only older revisions or obscure browsers might not support this level.

By executing the `yce_setup.pl` script and answering 'Y' to the 'SSL-hardening' prompt will the 'TLS1.2-only' access be configured on the system. A 'N' to this prompt will configure access for

TLS1, 1.1 and 1.2. The older SSL2 and SSL3 levels will always be disabled.

```
YCE server roles:
  #  Hostname         Front-end    SSL     HTTPD     URL      Mojo
Database     Id
* 0) yce-one          yes          https   root      name     8080
yes          1
  Select the server# to change, 'C' to continue: [0]
    'yce-one' is (also) a Front-end server? [Y]
    'yce-one' is DNS resolvable (y/n)? [Y]
    'yce-one' uses SSL (y/n)? [Y]
    'yce-one' uses SSL-hardening (y/n)? [Y] ?
      enter 'y' for TLS1.2 only, or 'n'
    'yce-one' uses SSL-hardening (y/n)? [Y] y
```

Single-sign-on

NetYCE servers support Single-sign-on (SSO) where the user can login on one NetYCE server and have login-free access to the other NetYCE servers. This SSO function worked as intended, but still was cause for confusion when 'a' NetYCE system would spontaneously force a logout or deny access to a tool due to 'permissions'.

The underlying cause was that customers use various NetYCE environments for 'production' and 'testing'. Each of these environments can consist of various servers but each has its own (replicating) database. A user logged-in to one environment can access a server in the other environment and finds that the session works - up to a point. This was caused by the fact that the user and his credentials exist in both environments, but the 'proof' of logging in, the session-id, is not present in that database.

The SSO function has now been redesigned to permit access across environments. The session-id does not have to be present in the 'guest' environment to be validated and accepted - provided the user-id exists in both environments and use the same password. If the users' permissions differ across the environments, they will be applied according the environment he is working in.

Reports

NetYCE offers custom reports using SQL queries on the various databases it manages. These reports of old were executed on the NetYCE server the user defined them on. Also the resulting reports were stored on the local disk of the server. Consequently the reports could only be viewed and retrieved on the same server.

When several servers are deployed the user is often aware where the report is available. However that is often not the case when the report is to be executed and/or retrieved using the API.

To allow transparent access to report definitions and the generated reports using the both the front-end and the API, the reporting back-end has been reworked to use the NetYCE database as the shared environment. Reports can now be created, edited and viewed from any NetYCE server and the API will execute the reporting from any server.

Generated reports can also be retrieved by URL directly from the browser. The URL uses the format `https://genesis.netyce.org/report/Report_name`. This results in a csv file in Unix format, but can be converted to Dos using `https://genesis.netyce.org/report/Report_name&type=dos`

The scheduled execution of the reports is balanced (by number of reports) over the available servers and still uses the standard Unix 'cron' to support the existing scheduling intervals.

Ping tool

A new 'ping' tool was build to replace the existing 'ping' and 'ping6' tools. Where previously only nodes fully modeled in NetYCE could be pinged, it can now ping modeled NetYCE nodes, CMDB nodes and ad-hoc devices using IPv4 and IPv6.

Ad-hoc devices can be entered using a fqdn or an IPv4/IPv6 address. The tool will ping all (resolved) ip-address three times reporting the resulting round-trip times in ms, or as a red '-' for each failed ping.

CMDB Management VRF

NetYCE transfers files between its servers and the Network devices by initiating the session from the devices. More and more designs (and vendors) require that these connections use the designated Management VRF and for properly modeled YCE nodes this VRF selection works automatically.

However for CMDB nodes where the available information on each device is severely limited, there was no option to select the proper Management VRF. To support these configurations the Cmdb_nodes now have an additional attribute to configure the 'Mgmt_vrf_name'. By default this attribute is empty signifying no management VRF is used. When a value for mgmt_vrf_name is inserted, it is used for specifying the VRF name for the transfer.

If a node name exists as both an YCE and and as a CMDB node, the YCE node configuration takes precedence. The Mgmt_vrf_name is strictly used for CMDB nodes. Basic command jobs for ad-hoc devices that have neither an YCE model nor a CMDB record cannot use a VRF.

Scenario csv-file/report commands

Two reporting commands were added to the scenarios: `csv_report` and `csv_file`. The csv_report can create a custom report output file from variable lists created in the scenario. A report is created from a single or a number of variables that have values that are related by row.

The first value of each supplied variable is combined into a row and converted into a csv (comma-separated-values) line. By doing the same over all items in the lists a report is generated. The 'csv_report' adds the result to the custom reports where they can be viewed and downloaded using the GUI, or fetched using the API or URL.

The 'csv_file' generates the same output data but saves it as a local file in the job directory. It is

then viewable and downloadable from the job details window.

To put this csv-file to additional use: the `send_email` command was for this purpose extended with the `-a <attachment>` option. This will allow the operator to create a log or data file and mail it to his desk, right from the scenario!

Change

backslash '\'-protection in scenarios

When using templates and scenario's, some characters have special meaning. These are used in templates to indicate a conditional configuration line (using `|..|`) or a sub-template to be substituted (using `{..}`). And in scenarios to indicate a special operator (like \s\d\r\n) in regex conditions (e.g. `/^\s*\d+/`).

To prevent these characters form being interpreted as a template operator and just type the character, backslash-protection is used. The string `{name}` refers to the template 'name' but the string `\{name\}` refers just to the string and will output `{name}`.

However, using this protection in scenarios would lead to unexpected results, often resulting in multiple levels of protection. The regex example above would work only when stored as `/^\\\\s*\\\\d+/`.

This less-than-desirable behavior has been addressed in such a way that only a single backslash-protection is all that is needed. The regex in a scenario is typed as intended without additional backslashes and to use a literal character in a template or scenario only requires a single backslash. The special characters like `\n` for a return or a `\t` for a tab are substituted where expected.

> **Important**: The modified processing of backslash-protection might result in some existing scenarios to fail. Multiple sets of backslashes (`\\\\`) will result in, indeed, backslashes (`\\`). These scenarios need to be reviewed and tested.

SSH-config

Most communication with the devices nowadays use the SSH protocol. This protocol evolved over time and now can use different versions using various ciphers and encryption standards.

NetYCE always used the default SSH protocols and ciphers that was installed op the operating system, and those defaults changed when moving from CentOS/RedHat version 6 to version 7.

To avoid no longer being able to connect to devices due to a change in supported SSH settings, the default SSH configuration is now overruled by a configuration file that will incorporate all available features of the SSH version running. Since this applies strictly to **outgoing** connections there is no security impact as it is the receiving device that determines the acceptable protocols.

The SSH configuration file is `/opt/yce/etc/ssh_config` and can be modified by the server administrator when desired. This configuration will only be used by NetYCE jobs, for cli-based sessions the default system SSH config file `/etc/ssh/ssh_config` is used.

Note: NetYCE uses the system SSHD configuration file `/etc/ssh/sshd_config` for its **incoming** connections.

Form permissions

When NetYCE denied the user access to a menu, form or tool, it was not always obvious that insufficient permissions were the reason for a lack of response or a prompt (re-login) to provide better credentials.

The various cases were evaluated and modified to provide the user with a consistent message reporting *“You do not have permissions to view this page”*.

Some inconsistent privileges were corrected.

Parsing Template test

When we introduced the command parsing with its own set of templates, we also added a tool to assist the engineer to create these command parsing templates. This 'Parsing test' tool was located in the 'Design' menu under the 'Template trace' option.

Because it was considered more logical to have it available as an operational tool, it was relocated to the 'Operate' menu under the 'Tools' option.

Scenario assignments

When the scenarios were extended with the support for variables, support for nested and indirect variables was included. This allowed for a very flexible definition of variables where even the order of the definition of a variable was properly 'resolved'. You could define for example `<if_name> := “Eth<slot>/<port>` and assign the <slot> and <port> values later, or even change them within a loop.

However, this 'feature' implied that the <ifname> variable did not contain what was expected resulting in failed tests values or 'saved' values that all of a sudden proved to be all of the same value!

Since these out-of-order value assignments are for most people counter-intuitive, we decided to make all assignments in the scenario (like <var> := <value>) to 'resolve' their nested and indirect values - if defined. The variable references that do not yet exist are maintained. The resulting variables will then contain their expected values and still support the out-of-order definitions if needed.

To properly support any as yet 'unresolved' variables referenced in an if-condition, these missing values are substituted for the comparison but do not alter the variable value permanently. This

allows to still benefit from any out-of-order definitions and loop constructions if desired or already in place.

Zip-code size

The field size for Zip-codes was with only 6 positions too narrow for the codes used in some countries. The Zip-codes now allow up to 15 characters.

Fix

F5 BigIP fixes

Several issues dealing with the F5 BigIP vendor module have been fixed:

- At login the 'tmos' shell was not detected
- Depending on version, the backup 'no-passphrase' argument was unknown
- When a login failed, the log did not show for with user name

Huawei fixes

A problem when saving a configuration for backup or NCCM purposes failed. This issue is fixed.

Cisco censoring

When a device configuration from the NCCM environment is shown on-screen, the NetYCE system will hide security sensitive information like passwords for operators lacking the privileges. This censoring mechanism uses keywords and patterns to find these phrases to be censored.

An update to the Cisco IOS and XE vendor modules was made to improve the recognition of snmp-server configurations for snmp-v3.

NCCM database restore

NCCM databases could fail to be restored where an error in compatible patchlevels was incorrectly reported. This issue was fixed.

Template AND conditions

In templates, conditions can be combined to create 'AND' conditions (e.g. '|<var1>||<var2>|'). When used to test on multiple variables without compare values as in this example, inconsistent results were produced if one of the variables had no value. These results also made the repeat conditions ('||' and |!|') unpredictable.

These issues were resolved.

Template comments

Templates can use several types of comment characters ('#”, '!', '–') that mark the beginning of a comment line. The comment lines in templates are dropped from the generated configuration, save the lines starting with the character used by the device vendor. These comment lines are included in the generated configuration.

These included comment lines may contain variables to be substituted to make the comment clearer. However, when a sub-template was 'commented out' using this method, the configuration would still include the parsed sub-template - with the initial line commented out.

This was fixed by only parsing only the vendor comment lines and protecting the template brackets by including backslashes ('\{...\}')

Update: The introduction of this vendor comment line handling was observed to introduce an unwanted side effect: Template blocks (lines ending in backslashes '\' form blocks and are parsed as a unit) with multiple templates would result in only the first template to be expanded, the remainder to be commented out. By reworking the multiline parsing and extending the comment handling resolved this issue.

Unexpected device prompts

Vendor modules using the device CLI sometimes are confronted with unexpected prompts. These prompts are handled automatically by accepting the default using an 'enter'. It was noted however, that when these defaults were not used and the CLI demanded a typed answer, the job would not properly timeout as was the case in previous NetYCE releases.

This issue is now fixed.

OS upgrade file-sizes

Over the years file and file-system sizes have grown. Since NetYCE stores the OS-upgrade file sizes and disk spaces in bytes, the limit of regular integer databases columns are no longer sufficient for storing the now common numbers.

The limit of 4.2 GB is now raised to 18 PB (yes, Peta-bytes) by redefining the relevant database columns.

Rebooting HP C7 devices

A problem where the scenario command `reboot` for HP Comware-7 systems would not properly reboot was fixed. Answering an additional prompt and a timing issue resolved the problem.

SFTP transfers

Most networking jobs will use SFTP as the preferred file transfer method. NetYCE uses a secured SFTP environment involving 'MySecureShell'. The configuration file for this environment, `/etc/ssh/sftp_config`, is created on server installation based on the distributed version.

It now has become apparent that this standard configuration file uses some performance limiting settings which allows only one session at the time at limited bandwidth.

An updated SFTP configuration file will be automatically installed when updating the software, but might require modifications on some systems. The new configuration limits the total SFTP bandwidth to the devices (download) to 100 Mbps with a max of 10 Mbps per session up to 50 parallel sessions. These settings were chosen to prevent poor performance for users when massive OS-upgades are in progress. See the Wiki article File-transfer Account setup for details.

Highest port-id

When adding ports, a maximum value of 256 for the port-id was used in the GUI. Plenty for ordinary devices but no longer for high-density devices. The same maximum value also applied to port-channel ports which supports values up to 4k.

These limits for physical and logical port-ids are now raised to 99.999. Since the node slot port-overview is limited to 150 icons an icon with '⇒' is diaplayed to indicate higher port-ids then shown are present.