Table of Contents

NetYCE 7.0.5 Build_20181203

Release notes

Date: 2018-12-03

Enhancement

CMDB objects

In preparation for the all-new 'Basic Command Jobs' currently under development, several reserved objects named 'CMDB' have been added to the NetYCE database. These basic command jobs will permit the creation of command-jobs for all devices present in the CMDB table. These basic jobs have full support for command and config parsing and have limited parameter substitution capabilities.

To support the use of basic templates and relations, the 'CMDB' objects for 'Client-type', 'Region', 'Client', 'Site', and 'Service_class' have been added. These objects cannot be deleted since they are essential for the functioning these basic jobs.

For environments where these 'CMDB' objects are perceived as undesirable, they can easily be hidden by removing the 'CMDB' client-type from the NetYCE user-groups.
Note: Hidden from the BUILD → MAIN only. The Objects are always seen under Operate → CMDB

New Topo check report

A specialized report has been added to the Operate - Reports menu: 'Topo check'. This allows the modeler to report on all topology links where a mismatch exists between the assigned subnets at both ends. Depending on the design such a mismatch could be an issue or not.

The output lists topos with mismatches only and is sorted by Client_type, Hostname, Link-type. Per topology link the missing subnet details are listed.

Change

Command jobs

The preparation and execution of command jobs was extensively modified with the 7.0.5 release. One of these changes involved generating the command job configurations at the time of execution instead of job submission time.

The scenario command '**Cmd_exec**' was modified to generate the commands from the 'template' that are the commands entered. It would use this <jobid>.cmd as template to generate the commands using the node specified in the '**-n**' option and then execute the result on the same device. The '-f' option is used to name generated command file for reference purposes.

The scenario line would then be similar to this:

```
-- example
<node> = NL-ACME-CPE

Cmd_exec -n <node> -f <node>.cmd <verbose>
-- generates NL-ACME-CPE.cmd using NL-ACME-CPE data and executes it on
NL-ACME-CPE
```

However, this approach would prevent you from generating the commands from this 'template' when the execution must be performed on a different node than the node (or context) these commands were generated for. To resolve this issue, the option '**-o**' was added to the Cmd_exec. The '-o <node>' (origin-node) specifies the node name whose context is used to generate the commands from the command-job 'template'. These commands are then executed on the node specified with '-n' as usual.

When the -o is omitted, the '-n <node>' is providing the context from which the commands are generated. The Cmd_exec will NOT generate the named file if it already exists.

```
-- examples
<node> = NL-ACME-CPE
<code_node> = NL-AMS-CR02

Cmd_exec -o <node> -n NL-AMS-CR01 <verbose>
-- generates the file NL-ACME-CPE.cmd using NL-ACME-CPE data and executes
it on NL-AMS-CR01

Cmd_exec -o <node> -n <core_node> -f <core_node>.cmd <verbose>
-- generates the file NL-AMS-CR02.cmd using NL-ACME-CPE data and executes
it on NL-AMS-CR02
```

The change requires the engineer to explicitly define the context and the target nodes instead of relying on a command file that was implicitly generated at job-submission time. This approach also allows to re-use the command 'template' for multiple nodes within the scenario. The context and target nodes can be switched as appropriate.

```
-- example
<node> = NL-ACME-CPE
<code_node1> = NL-AMS-CR01
```

```
<code_node2> = NL-AMS-CR02

Cmd_exec -o <core_node1> -n <node> -f <node>_part1.cmd <verbose>
-- generates NL-ACME-CPE_part1.cmd using NL-AMS-CR01 data and executes it
on NL-ACME-CPE

Cmd_exec -o <code_node2> -n <node> -f <node>_part2.cmd <verbose>
-- generates NL-ACME-CPE_part2.cmd using NL-AMS-CR02 data and executes it
on NL-ACME-CPE
```

Node port display order

When viewing the ports of a node, the ports are virualized per slot in groups of ports per port-type (e.g. Fast_ethernet, Gigabit-ethernet, Loopback, Port_channel). The order of these port-types was often inconvenient since it was more or less random.

This behaviour is now changed and lists the port-types in a fixed order: Ethernet, Fast_ethernet, Gigabit_ethernet, TenGigabit_ethernet, etc. The order is controlled using the internal Port_class table where also the port-type name is mapped to the port-class. If required, the order can be modified to customer taste.

Node port display truncated

On nodes where the port-ids of a port-type exceed 150 will the clickable ports now be truncated. This is most often the case for Loopback or Port_channel type ports, but the behaviour is general. It will prevent extensive scrolling past unused port-id's and improves the rendering speed of the ports by the browser.

When the list of ports for a port-type is truncated, it is indicated by a [⇒] symbol indicating the list continues but is not shown. The full list will be available using the grid at the top of the form.

Fix

Scenario variable resolving

With the redesigned command job and scenario handling of version 7.0.5 the variables handling was modified. The command generator now has access to all scenario variables which in turn extended to the relations.

With this behaviour in place we believed the scenario would no longer require variable substitution from the node. This assumption proved incorrect. Although scenario facilities exist to access relations for a node, existing scenarios in production were incompatible.

As a result the original scenario behaviour has been re-implemented. At time of execution, the entire scenario is parsed for variables and relations from the <node> context and substituted. This restores the functionality of the existing scenarios.

Node_log support

The Vendor modules were for the 7.0.5 release updated to behave more consistently and allow better control over error conditions. This was partly realized by creating more common functions for all vendors to use. Most of the existing functions were ported to these new vendor modules, but not all. The function to store the results of 'show' commands proved to be amongst these, although not intentionally.

To correct this situation, the function has been re-implemented. CLI commands starting with vendor specific key words like 'show' or 'display' are detected and their results stored in the YCE Node_log table for later examination or reporting.

- the 'show'-type commands must be fully typed, abbreviations are not supported
- the 'show'-type commands are executed as any other, only afterwards is the command evaluated as a storable result and saved in the database.
- the 'show'-type commands are expected to be executed in the appropriate mode. The vendor module will not magically switch between enabled and disabled modes.

Most vendors have the commands for 'show' or 'display' and 'ping', 'dir', 'do' recognized.

Duplicate node slots

A bug where romoved nodes did not clean up their node-slots caused duplicate node-slots when these nodes were re-created. Altough the problemof not removing the slots was fixed, the existing duplicates would not be removed until a port was added or removed from the node.

A patch will be executed when updating the NetYCE system to remove these duplicate slots.

Show command logging

When executing cli jobs on a device, the job can contain 'show' commands. The results of these commands are logged in the job log but were also stored in the NetYCE database in the Node_logs table for reporting purposes.

This functionality was not ported correctly to the various vendor modules when version 7.0.5 was prelim released. However, a bug prevented the actual use of this restored function which has now been fixed.

Command parsing

Command parsing in job scenarios fails due to a misconfiguration in the state-machine configuration of several vendors. The functionality has now been restored.