Table of Contents

NetYCE 7.0.3 Build_20171214 3
Release notes 3
Enhancement 3
Change 7
Fix 9

NetYCE 7.0.3 Build_20171214

Release notes

Date: 2017-12-14

Enhancement

Management-ethernet port-class

A new port-class 'ManagementEthernet' (internally named 'Me') was added to support the dedicated Gigabit-ethernet interfaces that some vendors integrated on the main board to support out-of-band management over Ethernet.

The class was added in generic format although the actual models were added only for the HP-C7 and Huawei vendor-types at this stage.

Active Directory

NetYCE can connect with Active Directory over LDAP for authentication purposes. The user-group can be retrieved using a basic attribute-based LDAP schemas.

This functionality has been extended to support more complex schemas where user objects contain 'memberOf' lists with roles that refer to application user-groups.

LDAP user attributes

For users logging in using LDAP authentication User records are created. These records can now be enriched with information from the LDAP user. Five NetYCE User attributes can be mapped to LDAP account attributes using the NetYCE Ldap setup records defined in the 'Yce_setup' table. These new user attributes belong to the 'Ldap_schema' section of the login profile and are optional. The following attribute names can now be added:

- 'usr_name_attr' - Name of the 'usr_search_base' attribute for the users 'Full_name'
- 'usr_mail_attr' - Name of the 'usr_search_base' attribute for the 'User_email'
- 'usr_notes_attr' - Name of the 'usr_search_base' attribute for the 'User_notes'
- 'usr_tel_attr' - Name of the 'usr_search_base' attribute for the 'User_tel'
- 'usr_mobile_attr' - Name of the 'usr_search_base' attribute for the 'User_mobile'

NCCM archives

The NetYCE database NCCM was split from the YCE database in version 7.0.1. The NCCM archiving tool allowed the manual creation of an NCCM archive set, but was not created daily.

Automatic daily archives of both YCE and NCCM are now standard and takes place at 23:00 or 23:20 (primary or secondary server) by default.

The maximum number of archives for each type are now also controlled using Tweaks in the Lookup form. The '**Archive_count_yce**' variable defines the number of historical YCE database archives that are maintained. Likewise the '**Archive_count_nccm**' defines the number of NCCM archives. Both use defaults of '15'.

All archives of each type count against these values, regardless if they were created manually, uploaded, or created automatically. The only archives NOT counted (and therefore not deleted) are uploaded archives where the filename was modified.

The deletion of excess archives takes place after each archive creation, regardless of type.

Service-types 'Locate-Service-<client>/<site>'

Service-types support for locating a service has been extended to locate the service in a client-alias or a service-alias. The match criteria for the service include Service-type, -name and -class.

With the addition of these six new syntaxes, there are now 19 different options to locate a service.

Templates for vendor-type '-any-'

Port-templates and Sub-templates can be defined for a specific vendor-type, but also for *all* vendors using the vendor '***-any-***'. When the system selects the proper template for a device, the template matching the devices vendor-type will have precedence over the '-any-' vendor thus allowing for generic, vendor independent, templates. Bit it also allows to use default templates that can be overruled when a vendor-specific version is created.

The '-any-' vendor, like any other, also supports 'planned' template revisions. For testing purposes the 'View config' tool allows to use these planned revisions where available. In this case the precedence order becomes: 1) vendor-planned 2) '-any-'-planned 3) vendor-production 4) '-any-'-production

The '-any-' vendor type was introduced with the release of NetYCE 7 but was not available for configuration generation until now.

Custom subnets using Service-types

When adding new CUSTOM subnets to a service using a service-type, the subnet is created with the Net_name provided. The Net_addess, Net_size and Net_mask needed all be specified using additional service-type commands, one for each attribute.

A simplification has been provided where the Net_address can include the Net_prefix using the notation “<address>/<prefix>” from which the various attributes are calculated.

If the service-type is intended to be used over the API, this same format can be used by the Add-Subnet-Custom command by using a custom variable. If you add the 'custom' variable 'Net_address' in the API call using the <addr>/<prefix> format, the Custom subnet is created using the attributes derived from this value. In addition, the Add-Subnet-Custom also supports the API 'custom' variable “Net_size” (or “Net_prefix”) to set its attribute value.

Synchronization repair

When using redundant NetYCE databases in a master-master configuration, the databases are kept in synchronization using a sophisticated replication mechanism. This mechanism involves temporary replication log files and a pointer administration on each server.

This mechanism can fail in two distinct cases. The first is where the servers' disk file system runs out of space to host the growing replication logs. The second is where one of the database table files gets corrupted. Observed cases of table corruption always involved heavy usage where a great number of records were added and removed. The corruption is then created during a daily cleanup where the table fragmentation causes the datafile to be rebuilt. During this process, any filesystem issue will cause a corrupt file. Usually buy repeating the process the corruption is auto-repaired - if disk space suffices. This leads to the conclusion that insufficient free space can cause severe database problems.

Once these problems manifest - database synchronization failure - two options are available. For situations where one server can be identified as having a healthy and complete 'master', the standard synchronization procedure using the double archive can be used.

The other situation where the databases really need to be repaired and records must be merged, only a manual recovery procedure is available. This procedure is to be executed by a NetYCE application manager. NetYCE incorporated a database recovery tool for this purpose some time ago (ck_dbsync.pl), which is now extended to include synchronization-reset options. This resulted in an extensive and flexible procedure. The procedure can be found on the NetYCE Wiki: NetYCE Database Re-synchronization procedure

A more general description of the database replication is given in: Database Replication

Browser cache

With the release of YCE 6.3 we adopted the 'appcache' mechanism of controlling the caching capabilities of the various browsers. This is a method where the webserver and the browser use an application specific configuration file on what files and under what circumstances the webserver and the browser may cache files. At the time, the 'appcache' was gradually supported by the modern browsers.

Unfortunately, its implementation in the various browsers and their many os-variants proved that the 'appcache' became increasingly ineffective. The result was that with every revision or patchlevel update of NetYCE, many users were forced to manually clear their browsers cache. An option that was also increasingly hard to locate.

To alleviate this problem we decided to continue the appcache support, but revert back to the old-and-trusted method of including versioning information in the URL's requiring it. This method provides the developers a means to specify which files must be replaced in the browser cache on a version or patch update. It is a crude but effective solution to an issue that often caused irritation by the users.

We added into this method an additional mechanism that forces a cache refresh for users that were actively using the system during the NetYCE software update. From the moment a new version installation is completed, the very next action the user completes will result in a page reload (and partial cache refresh). All the user will notice is that he is 'spontaneously' directed to the main Build form.

For users that have bookmarked the old NetYCE login page, a popup is displayed to inform the user that he will be redirected to the new login page and his bookmark needs updating.

GUI support for tweak 'Allow_topo_multilink'

The recently added tweak 'Allow_topo_multilink' was limited to the Service-types. This functionality has now been extended to include the NetYCE GUI. With the tweak enabled for the Client-type at hand, multiple topology links per port can now be created.

Node_type substitutions

When defining a Node_Type, a number of records require entries that will be assigned to any new node using that node-type. These values can be literal text or the output of a function.

To simplify and expand the flexibility of these value assignments, many of the variables related to the new node can now directly be used in the value using the same notation as the variables in the templates. The variables in pointy brackets ('<...>') are replaced by the values available at that point in the node definition.

Where previously about 10 variables could be used in this fashion from within functions, now all Client, Site, Region, Service and Domain variables including any custom attributes can be used directly. Previously the' function' Concat() had to be used to create a string that used variables, now the string can be typed as a 'literal' where the <...> variables will be substituted. The existing Node_types will function as usual.

Service-type: Locate-Address-Address_lastfree

To deal with designs that stipulate that individual addresses from a subnet range need to be assigned from last through first rather than first through last, the service-type 'LOCATE - ADDRESS - ADDRESS_LASTFREE' has been added and is complementary to 'LOCATE - ADDRESS - ADDRESS_FIRSTFREE'.

Likewise the service-type 'LOCATE - IPV6_ADDR - IPV6_ADDR_LASTFREE' was added.

API authentication

The NetYCE API authenticates each request using a username and password. That password may be provided in three formats: in cleartext, in MD5 hash and in encrypted format. The detection of the format is automatic.

When LDAP or AD is used for user authentication the user password was validated once against the LDAP server and then thrown away. This setup posed a challenge when using the API as there was no means to authenticate the user at that stage.

To resolve this issue, the LDAP password design is modified in such a way that the LDAP passwords are maintained in the NetYCE user administration along with the local passwords. This enables the API to validate the request authentication locally (not checking with LDAP for every request).

There are two consequences of this arrangement. First is that when authenticating against LDAP, the user's local (and API) password will be updated to the password used by LDAP. User accounts that are setup to be 'local' but can also use LDAP, will find that the 'local' password is now that of their LDAP account too. Secondly, LDAP users must have logged in at least once to the NetYCE front-end using their (new) LDAP password before submitting API requests. The LDAP password gets only included in the administration using the front-end login procedure.

Change

Database Synchronization

For NetYCE database servers using the master/master synchronization option, a change in the master-slave setup has been incorporated that is more reliable. Instead of filtering updates at the master to be sent to the slave, now the master sends all actions and the slave determines which tables to replicate.

This replication change MUST be executed with minimal delay between both NetYCE database servers to permit the synchronization to continue.

This change will also include replication updates to its (new) NCCM database and the (future) CMDB database. Any existing NCCM database must be restored at both servers to start synchronization.

Domain password lengths

Passwords maintained in the Domain table are by default encrypted. This encryption requires more storage space than unencrypted entries. The Domain table used column widths of 100 characters to allow for these longer values.

However, when storing password values in a format that is not clear-text but uses device-specific encoding or encryption, the required storage space of the database-encrypted string exceeds in some cases these 100 characters.

For this reason, the storage space for passwords in the domain table has been extended to 200 characters.

Maximum Password length

The maximum user-password length has been increased from 20 to 50. This is to facilitate users wishing to use password phrases. Longer passwords will be ignored at login time.

These 50 characters is the absolute maximum allowed by the system, the administrator can reduce the maximum length by updating the Lookup entry for 'MaxChars' in the 'Password' class.

Scenario Db_update command

The scenarios have a command available to change a value in the database, Db_update. This command was redefined for version 7 so that it could change multiple variables in the same database table. The original could set only one variable.

To support multiple variables the Db_update command options were changed where the '-f “field”' and '-v “value' options were dropped in favour of multiple '-p “field=value”' options. This change required existing scenarios te be modified which was not obvious to many users.

A modification to the Db_update command has been made by which the new '-p' options AND the original '-f' and '-v' option set is supported.

One point of attention remains for existing scenario users: Option attribute values need to be enclosed in quotes when spaces are included. They are essential in much more formal scenario parsing of version 7 needs it.needs it.

Custom Data Grid

The custom data grids now are twice as big, and the last entry is now fully visible.

Subnet assignment to Ports

When assigning subnets to interfaces using the front-end GUI or the service-types, the subnet's ip-plan is used to determine which ip-value from the subnet is to be applied to the interface. This selection depends on the subnet's function and any topology this interface may have.

Over time, various subnet-names have been hardcoded to be used for point-to-point or loopback usage in the front-end but not for service-types. The resulting mappings gave inconsistent results.

To rectify we changed the behaviour to that the assignment soley uses the subnet-plan details and

the topology. The subnet-plan has individual settings for ip-address calculation in three modes: 'loopback', 'point-to-point', and '4-corners' (North, South, East, and West references). Based on the presence of these settings in the subnet-plan the correct ip reference is now determined for both front-end and back-end usage.

An accompanying NetYCE patch (17120601) will completely recreate the subnet-to-port mapping in the database to make it consistent.

System page

The 'Admin - System' tools have been the premise of the users with 'manager' level permissions. However, the 'Admin - System - Edit configs' tool has some configurations that are of interest to 'modeler' level users too.

To make this tool accessible, the 'System status' tool has now been modified to allow 'modeler' level user the basic permissions to browse the page, but are not allowed to make changes.

Fix

Infoblox export

The Infoblox exporter failed intermittently to export dns records for a set of specific dns zones. The problem was introduced when the 'ibd' daemon was reworked to report by Network-view. The Dns-views residing under each Network-view are then reported zone-by-zone.

As it turned out, the zones from one Dns-view had a 'forwarding' view in another, but only the 'authoritative' zone has any data. Since the order of the reporting was random, the empty results of the forwarding zone would result in an empty dataset for the zone.

By skipping the 'forwarding' type zones, the exported dns records are consistenly complete.

Installation package

A recent enhancement was aimed at relauncing the various NetYCE daemons at the completion of the update process, but before executing the update patches. This 'relaunch' caused indeed all daemons te be restarted, including the API process that was updating the system.

As a result, the update progess indicator (issuing a growing list of '.') to stop and never returing the update log to the user. In the background the update process still completed the operation, including the update patches.

The relaunching of the various daemons now skip the API causing the update to complete the process properly.

Archive timeout

When executing database archives or restores of very large databases the users Browser would timeout while waiting for the process to finish. When this happened the archive or restore procedure would be aborted, usually leaving the database stopped or patches partially applied.

Now the user is informed every five seconds how much time has passed waiting for the current operational step. This is both informative for the user, but it also keeps the Browser from timing out and thus ensuring the operation will be completed.

API Add-Supernet-Ip_plan

When adding a supernet to a client using a service-type (Add-Suppernet-Ip_plan), the action was aborted with “ERROR Failed ADD - SUPERNET - IP_PLAN: IP-plan '123' not found for client_type 'AA'”.

The problem was an error while resolving the available ip-plans for the client-type where the client type was left blank. This was resolved by correctly referencing the client-type.

OSS Integration parsing error

Using an array of datasets with various network-footprints, the parsing and variable validation produced unexpected results. The problem was caused by referencing the various network-footprints in random order. Each dataset is processed in sequence, each time loading the appropriate configuration file for the footprint.

However, when the same footprint was again referenced for another dataset, the system would ignore the config-file load, assuming it was already loaded. The result was that the dataset would be processed and validated using the last original configuration file. In a random ordered dataset that would lead to erroneous results.

The issue was resolved by maintaining all configuration file loads internally, thus eliminating the dependency on the repeat usage of the same configuration file (and associated procedures).

IP Plan Create

Fixed a bug where you could not create an ip plan after deleting it.

Subnet VRF

You can now set IP Subnet VRFs to empty again.

Node deletion leftovers

When deleting a node, either manually or using a service_type, some database records were not removed causing (harmless) clutter in the Ip_map table. This has now been fixed.

Non-existing Port subnet assignment

The topology form of any node would display at least one, incomplete, entry in the port subnet list grid. Regardless if any subnet was assigned to the selected port, one entry would always be listed. It proved that this phantom entry was due to an imprecise sql query that feeds this grid. This has now been optimised.

If incomplete entries are now listed in the grid, these actually represent incomplete subnet references. By selecting the checkbox “Connected nets only”, these can be filtered out as well.

Custom Data

Custom Data views now all have pagination. You can search and your search will be on the entire dataset. Sorting only works on the current page. All IPv6 fields are hidden now, since viewing and editing them falls beyond the scope of what Custom Data was meant to be.

Node bootimage reset

Resetting the boot loader in the node edit form also updates its value in its form.

Node Redundant

The node edit form now correctly displays if a node is redundant.

Menu buttons

Some menu items only were selectable if you clicked on their text, rather than their menu item. They now respond to the whole menu button.

Subnet Topology not set

When assigning a subnet to an interface using the front-end, the topology position of that interface in the subnet would be set to '1', instead of the familiar 'NA','ZB',etc. This behaviour is corrected. Previous incorrect assignments are corrected using a patch.

Switchport config wrong node

The issue where the switchportdetails form showed the different node when you open the form through the services window is now fixed.

Node redundant fixed

The issue where the Redundant option for a node got saved as NULL has been fixed. Also includes a patch that changes all NULL-entries in the database back to 0.

Node Class Notes

Fixed the issue where you get prompted repeatedly when switching editing node classes without saving.

IPv4 Subnet Forms

Fixed an issue where vlan templates were missing when creating IPv4 subnets, and the correct values not updating.

Custom attributes group sorting

Fixed a bug where the grouping sort suggestions for the custom attributes form displayed the wrong possible values.

Changing Job statuses

With the introduction of version 7.0.2 a periodic check was introduced to detect Jobs that were reported 'Runnng' but were actually finished. Prossibly due to process crashes. These job states would then be modified to 'Aborted'.

However, the implementation examined the 'Running' jobs of all NetYCE servers, but compared the existance of the job only to the local server. As a consequence, one server would conclude the job failed while it was still running on another server. The user would then observe that the running job would change into an aborted one which could then change into a successful state when it finished.

The job status monitoring is now confined to local jobs only.

Service type records

Creating a new service type record now immediately jumps to that record automatically, and you

don't have to manually scroll anymore.

Confirm window enter

A confirm window (the windows with a simple message, ok- and cancel-buttons) can now always be closed with enter.

Job log details

Job details of a remote server whould not be shown. The result window of a job on a different than the local server would open, but no data was retrieved. This was corrected.

Vrf_id 'null' values

The Vrf_id column was defined in the various tables using different default values. The Ip_subnet tables allowed Null entries were others use empty string ('') defaults and denied Nulls.

Since this was introduced in NetYCE version 7.0.0, it caused existing relationships to fail their criteria. Now all Vrf_id references use the empty string as default and deny Nulls.

Template variable typo's

When a variable used in templates and command jobs is not properly enclosed in angled brackets ('<' and '>') because one is missing, the view config tool and the command-job evaluate tool would not detect it and show nothing or just the name as plain-text.

The generated configuration will contain the literal string (the caret and variable name) as is intended. To detect and highlight these (probably) unintentional typo's, the above tools will now always display the variable name and show the unmatched caret in red.

Cisco-IOS 'Invalid encrypted password'

The Cisco-IOS error message concering 'Invalid encrypted password' went undetected. This string is now added to the Cisco-IOS error message list.

From:
https://yce-wiki.netyce.com/ - **Technical documentation**

Permanent link:
https://yce-wiki.netyce.com/doku.php/maintenance:releases:7.0.3_20171214

Last update: **2019/12/23 16:13**