Infoblox Extended Attributes mapping

For the Infoblox IPAM integration.

To create and maintain the full IPAM tree of a Client in the Infoblox GridMaster, the toll “IPAM / DHCP update” tool is installed when the Infoblox license is present.

This tool allows to select a Client and update or renew the entire IPAM tree of the supernets assigned to the Client. Since each supernet is divided into subnets of various types and size for specific purposes, the IPAM tree can be organized following this structure.

The structure can be explicitly defined per plan-id in the NMS table Dhcp_tree. The usage of this table is described in the article Infoblox IPAM / DHCP tree definition.

This article describes how the retrieved information for the definition of the IPAM tree can also be used to define the values of the Infoblox Extended Attributes.

These extended-attributes (or ext-attr for short) are highly customizable and fully customer specific. The Infoblox integration with NetYCE is therefore likewise highly customizable.

Available YCE attributes

The aforementioned IPAM / DHCP tool uses a three-staged process to manipulate the IPAM. First, a request is made to the NetYCE XML API to retrieve the information to create the desired tree. The response combines the Client's subnets as deployed in the NetYCE abstract network and the Dhcp_tree definition for the Ip-plans involved. Supernets of an Ip-plan that does not have a corresponding Dhcp_tree will not be included in the response.

The resulting information includes entire hierarchical IPAM trees that is composed of three types of objects: containers, networks and scopes (aka ranges). The attributes reported for each of these object types differ. The ext-attr mappings that can be defined are restricted to these attributes.

The second step involves adding automatically the desired Infoblox Extended attributes based on the information received. For all 'network' and 'scope' objects in the IPAM supernet trees these are 'mapped' to the NetYCE attribute values or literal values, again based on NetYCE attribute values.

The final step is creating or updating the Infoblox IPAM trees corresponding the retrieved and generated data.

Since the ext-attr 'mapping' is based on the NetYCE attributes of these objects, the set of available attributes was doubled to allow for extensive mapping schemas. The available set for 'network' and 'scope' objects is near-identical: the scope includes the 'scope_start' and 'scope_end'.

“network” Attribute	**Example: in-use**	**Example: free**	**Description**
net_type	**network**	**network**	Infoblox object type
line_number	6	6	sequence number in tree definition
net_tier	2	2	relative hierarchy level of object in tree
scope	IPAM or DNS	IPAM	name of mapping configuration. `IPAM` or `DNS`

“network” Attribute	**Example: in-use**	**Example: free**	**Description**
source	netyce, hpoo or linux	netyce	origin of requester. Currently only 'netyce' for IPAM. For DNS, netyce, hpoo and linux are valid.
task_id	1214_0001	1214_0001	NetYCE API's unique transaction id. Format <mmdd_nnnn>
client_type	NY	n/a	Client_type
client_code	ASD-DC	n/a	ClientCode
site_type	DC		Site_type. Type of location. When network is not assigned to a location or service, this value is blank
site_code	ASD–NY01	n/a	SiteCode. Location code
service_type	L3_AccessU-48	n/a	Service-type of the service containing this network
service_name	L3_AccessU-48		Service-name of the service containing this network. The name is equal to the Service-type by default and is user-alterable
net_address	10.10.0.0	10.10.1.0	Ip-address of the network
net_mask	255.255.255.128	255.255.255.128	Net-mask of the network
net_size	25	25	Prefix or CDIR of the network
net_name	Users	Users	Name of the subnet-type as per ip-plan
net_descr	Users	Users	Name of the subnet. Equal to the net_name by default and is user-alterable
net_comment	DC - ASD–NY99 - vl401 - Users 0	Users 2	This string is entered by default in the Comment. Value is hardcoded
vlan_id	401	n/a	Vlan number if assigned one
vrf_id	12	n/a	VRF id number this network was assigned to
vrf_name	ka		VRF name, according NetYCE VRF definition, this network was assigned to
ddns	yes	no	Enable dynamic dns: 'yes' or 'no'
net_options	1,15,44,46,51	1,15,44,46,51	List of dhcp-option numbers that are configured for this network
“scope” Attribute	**Example: in-use**	**Example: free**	**Description**
net_type	**scope**	**scope**	Infoblox object type
line_number	7	7	sequence number in tree definition
net_tier	3	3	relative hierarchy level of object in tree
scope	IPAM or DNS	IPAM	name of mapping configuration. IPAM or DNS
source	netyce, hpoo or linux	netyce	origin of requester. Currently only 'netyce' for IPAM. For DNS, netyce, hpoo and linux are valid.
task_id	1214_0001	1214_0001	NetYCE API's unique transaction id. Format <mmdd_nnnn>

“network” Attribute	**Example: in-use**	**Example: free**	**Description**
client_type	NY	n/a	Client_type
client_code	ASD-DC	n/a	ClientCode
site_type	DC		Site_type. Type of location. When network is not assigned to a location or service, this value is blank
site_code	ASD–NY01	n/a	SiteCode. Location code
service_type	L3_AccessU-48	n/a	Service-type of the service containing this network
service_name	L3_AccessU-48		Service-name of the service containing this network. The name is equal to the Service-type by default and is user-alterable
net_address	10.10.0.0	10.10.1.0	Ip-address of the network
net_mask	255.255.255.128	255.255.255.128	Net-mask of the network
net_size	25	25	Prefix or CDIR of the network
net_name	Users	Users	Name of the subnet-type as per ip-plan
net_descr	Users	Users	Name of the subnet. Equal to the net_name by default and is user-alterable
net_comment	DC - ASD–NY99 - vl401 - Users 0	Users 2	This string is entered by default in the Comment. Value is hardcoded
vlan_id	401	n/a	Vlan number if assigned one
vrf_id	12	n/a	VRF id number this network was assigned to
vrf_name	ka		VRF name, according NetYCE VRF definition, this network was assigned to
ddns	no	no	Enable dynamic dns: 'yes' or 'no'
net_options	3	3	List of dhcp-option numbers that are configured for this network
scope_start	10.10.0.6	10.10.0.6	First ip-address in the scope range
scope_end	10.10.0.126	10.10.0.126	Last ip-address in the scope range

Mapping Configuration

Once the Infoblox extended-attributes definitions are finalized and implemented, the Infoblox ext-attr configuration file can be created. This configuration file is to be created as `/opt/yce/etc/`**`ib_extattr.conf`**.

The format for this file uses a pseudo-language to simplify the syntax by preserve the hierarchical nature of the configuration. The structure is outlined below:

```
# EXTENDED-ATTRIBUTE MAPPING CONFIGURATION
#
# scope {                                  # IPAM or DNS
#    Extended-attribute-name {             # Infoblox Extended-attribute name
#        source {                          # netyce, hpoo or linux
#            primary-key-attribute {       # mapping list supporting regex and
indirect values
```

```
#                key-attribute-value = value
#                key-attribute-value = <attribute>
#                key-attribute-value = pre<attribute>post  # attr are
substituted. Only 1 attr.
#                /regex-match/ = value
#                /regex-match/ = <attribute>
#                'else' = value/attr     # when no key-attribute-values
matched
#                'default' = value/attr  # when key-attribute is blank
#            }
#            secondary-key-attribute {   # optional, additional key-
attribute with mapping list
#                key-attribute-value = value
#            }
#            'else' {                    # optional, when no value was
obtained
#                key-attribute-value = value
#            }
#        }
#    }
# }
```

An example may serve best to illustrate its usage. Consider the definition of the IPAM ext-attr `Referencecode`:

```
# Example:
# The ext-attr 'ReferenceCode' will be assigned a value based on the
attribute 'client_type':
   Referencecode {
       netyce {
           client_type {
               default = missing-client_type
               rn = <client_code>
               /f+p$/ = <site_code>
               else = <client_code>
           }
       }
   }
```

When the NetYCE attribute `client_type` has the value 'RN', the ext-attr ReferenceCode is set to the value of of the the `site_code` attribute of the network.

But when client_type matches the regex /f+p$/, like 'FP', it will use the value of the `site_code` attribute.

Should neither match, the `else` defers it to the `client_code` attribute. The `default` case can be added to catch the situation where the `client-type` value is missing or blank.

In these mappings, right-hand side of the = can use literal assignments (value), attribute assignments (<attr>), or combinations. The assignment = Y15<`task_id`> is valid and will yield a

value like `Y151217_0023`.

This example uses a request for an an IPAM record (the scope) where the requesting application is 'netyce' (the source). Only 'netyce' is currently doing IPAM configurations for Infoblox, when other sources are added, create mappings for these as well.

More complex mappings can be created when several key-attributes are stacked together. Two or more key-attributes, each with their own set of mapping entries, allow first one attribute to mapped, and in case of no match, a second set and so on. The set can be extended with an `else` key-attribute to catch a 'no-match-found'. See the example below:

```
    Netwerkomgeving {
        netyce {
            vrf_name {
                /^ka/ = dn
                kn-vrf = dn
                /linux/ = <vrf_name>
                else = <vrf_name>
            }
            subnet_type {
                /oracle/ = dc
                /wifi/ = dn
                users = dn
            }
            client_type {
                fp = dc
                rn = dn
            }
            else {
                default = missing-vrf
            }
        }
    }
```

> **Notice**: Currently only the IPAM 'scope' is used. The 'DNS' implementation will be added later.

Notes:

None of the strings need to be quoted. String-enclosing quotes will be ignored.

Much of the configuration file is case-insensitive, but key-attributes and mapping attributes should be LOWER-CASE.

The mapping comparisons are always case-insensitive. `ny = <client_code>` matches 'NY' as well as 'ny' and 'Ny'.

The regex-support for the mapping entries is indicated by a regex between slashes ('/ ... /'). Regex modifiers like `/../i` will prevent the regex to be recognized.

Example configuration file

filename: /opt/yce/etc/**ib_extattr.conf**

```
IPAM {
        CI {
                netyce {
                        client_type {
                                fp = TI000456
                                ny = TI000123
                                default = missing-client_type
                                else = TI000789
                        }
                }
                mon {
                        ci {
                                default = missing-ci
                                else = <ci>
                        }
                }
                linux {
                        ci {
                                default = missing-ci
                                else = <ci>
                        }
                }
        }
        RFC {
                netyce {
                        rfc {
                                default = <task_id>
                                else = <rfc>
                        }
                }
        }
        ReferenceCode {
                netyce {
                        client_type {
                                /f+p$/ = <site_code>
                                ny = <client_code>
                                default = missing-client_type
                                else = <client_code>
                        }
                }
        }
        NetworkEnv {
                netyce {
                        vrf_name {
                                /linux/ = <vrf_name>
```

```
                                /^ka/ = dn
                                else = <vrf_name>
                                kn-vrf = dn
                        }
                        subnet_type {
                                /oracle/ = dc
                                /wifi/ = dn
                                users = dn
                        }
                        client_type {
                                fp = dc
                                rn = dn
                        }
                        else {
                                default = missing-vrf
                        }
                }
        }
        Source {
                netyce {
                        source {
                                netyce = NetYCE
                        }
                }
        }
}
DNS {
        CI {
                netyce {
                        client_type {
                                fp = TI000456
                                ny = TI000123
                                default = missing-client_type
                                else = TI000789
                        }
                }
                mon {
                        ci {
                                default = missing-ci
                                else = <ci>
                        }
                }
                linux {
                        ci {
                                default = missing-ci
                                else = <ci>
                        }
                }
        }
        RFC {
                netyce {
```

```
                        rfc {
                                default = <task_id>
                                else = <rfc>
                        }
                }
                mon {
                        rfc {
                                /^T000{\d}4/ = <rfc>
                                /^C000{\d}4/ = <rfc>
                                default = missing-rfc
                                else = invalid-rfc
                        }
                }
                linux {
                        rfc {
                                default = missing-rfc
                                else = <rfc>
                        }
                }
        }
        Source {
                netyce {
                        bron {
                                else = NetYCE
                        }
                }
                mon {
                        bron {
                                else = NetYCE for ItShop
                        }
                }
                linux {
                        bron {
                                else = NetYCE for Linux
                        }
                }
        }
}
```