Running Service-types

Th first thing to note about Service-type is that they are strictly limited to the NetYCE database. Service-types create, modify, and delete database objects. Nothing else. They prepare the data so that later it can feed the templates and the scenarios to execute jobs.

the "create" task

The first use of Service-types are for creating new services and/or nodes. In this case the Service-type MUST be created using the keyword **`create`** for the Service_task name.

Every Service type needs four identifiers to be unique:

- the `Client_type` to define the network architecture (or 'tennant'),
- the `Service_class` which is a 'class-of-service' that the `Site_Type` supports,
- the `Service_type` which is a specific service-variant to the `Service_class`,
- the `Service_task` which describes what the Service-type will do.

So the reserved `create` as Service_task is used to filter the these service-types when we want to create a new service to a clients' site using the front-end GUI. It is used mostly to create a new service-container in which a node is added and possibly some subnets.

the other tasks

Other Service_tasks can now be created to define standardized changes or perform migrations. Since the service was created using a Service_type name, these corresponding tasks can rely on knowing exactly the intended design was and operate on it.

Over time these tasks will be created, used and when obsolete, deleted. Since these tasks did not 'create' the service or node, the initial service-type commands are needed to 'LOCATE' the correct objects before they can be manipulated. For all object types service Service-type commands exist to locate an object given some criteria.

the "current" scope

For Service_types/Service_tasks that are executed using the GUI, the operator already provided some context by selecting the Client, Site and Service. Sometimes even the Node or Subnet. In those cases the Service-type can use the CURRENT-scope variant of the Service-type command, making the locate that more easy.

the API variables

Service_types/Service_tasks can also be launched using the NetYCE API's. There is the XML based XCH API and a simpler CSV-based 'wrapper' for this XCH API. Using the GUI, the Service-types have either to be provided with fixed values or rely on proper modelling for the actual value to use in the various

commands.

However, for the API it is possible to include “custom” variables and values in the XCH call. These variables are then inserted in the appropriate “value” places of the commands. To identify the variables in the Service-type commands, the user types the expected API “custom” variable name in the place of the “value” in the command. To distinguish between API variable names and literal values, the “custom” names must be entered between brackets: **`(custom_name)`**. There is no limit on the number of custom variables that can be included in the API or the Service-types.

Some of these “custom” names should be used with caution. Several Service-type commands can use “hidden” arguments in their execution simplifying the operation by not first having to create objects with defaults alone. Please consult the article on Special names.

filtering GUI tasks

To prevent executing Service_tasks that depend on API “custom” variables, the GUI will filter Service-types that use the `(custom_name)` reference in any value.

Rollback on error

In the chain of commands, something can go wrong. In that case, all modifications on the database are rolled back and undone, so that you don't need to clean them up. This a new feature, available from version 8.2.1