Relations

Relations can be used in templates and command jobs, and they enable you to extract information from the database. An extensive set of built-in relations are provided with the NetYCE installation, giving the engineer the option to build generic templates and jobs, but you can also create your own custom ones.

Relations are primarily used within Templates but can be used with great effect in Scenarios and Stored jobs too. Access to relations is NOT restricted by Client-type as are the Templates.

Relations are in essence MySQL queries that allows the builder to create temporary datasets based on NetYCE database tables and conditions. These datasets are generated on the fly during Template execution where their data can be used as variables.

The **Relations** form allows you to create, import, export, modify and delete your own Relations. The built-in Relations cannot be modified (unless you are a System level user) but adaptations are possible using a 'duplicate' relation. To access this form you need 'Modeler' (user-level 4) privileges.

The Relation test tool can be used to verify the relation and its results for a given node. To access this tool you need 'Engineer' (user-level 3) privileges.

How the relations are used within templates is described on the Relations and Contexts page.

Import/Export

Since Templates, Scenarios and Stored jobs will have dependencies on your Relations, the option to export and import created relations to other NetYCE (production) systems is as relevant as exporting templates.

Click an existing relation and then press the  button to export it. This prompts the user to store the scenario as `Rel-<scenario name>.xml` as the file name.

To import a relation, Click on the  and when prompted, provide the exported xml file. The file is validated and you are presented with a summary of import as shown below.

If the relation already exists, NetYCE will warn the user, and the `Overwrite` checkbox needs to be checked for the existing relation to be imported.

Dependencies

Relations can be used used in Templates, Scenarios and Stored jobs. To prevent deleting a Relation that is in use in any of these applications, a dependency check is performed when attempting to

delete a Relation.

The results of the dependency check is a detailed report on which templates, scenarios and jobs include a reference to the Relation and on which line number. The line with the reference is incorporated.

Depending on user permission level, the dependencies that prevent deletion can be overridden. If permitted, the dependencies report concluded with the message “Do you want to delete anyway?” and offers the choices 'Cancel' and 'Ok'.

The permissions are defined in the YCE.Auth_permissions table and can be changed using the 'delete_override' actions for 'relations', 'templates', 'template_revisions' and 'scenarios'. The default is that only 'Manager' and 'System' level users have the override permissions (Role_5 and Role_6 columns).

As NetYCE uses some pre-defined Relations that have system dependencies, these Relations are made read-only for all but 'System' level users. Those Relations may not be deleted even when no dependencies are reported. An (additional) “read-only' notification is then displayed. This behaviour is controlled by the column “Relation_edit” of the YCE.Relations table. The default is '4' (modeler) for customer-defined relations and '6' (system) for NetYCE provided relations. A value of '7' can be assigned to relations that should never be allowed to be either edited or deleted.

Variables

Variables within the Relations' SQL query are possible and flexible. There are several examples in the built-in relations which show the use of <Hostname> or <Port_name> like the example below (it extracts the interfaces of a node).

```
SELECT Port_map.*
FROM Port_map
WHERE Port_map.Hostname = '<Hostname>'
ORDER BY Slot_id, Port_name
```

Any variable known at the moment of execution can be used within the Relation. This includes all of the variables of the node, service, site, client, domain, region, etc that make up the normal 'context' of a node. But also using your own custom variables can be used in a Relation if you defined it in the [parameter] section of jour job.

This is especially useful in command jobs, where variables can be set as parameters.

Transformed columns

When creating a named context (relation) the SQL definition of the relation can be modified to automatically create additional columns based on the values found in the SQL results.

This extension of columns is the result of transforming a field value from a selected column into a column name. The value(s) in that column then points to another selected value from the record. In that way, a result table in which every line describes a parameter, can be changed to a table in which this parameter-name becomes a field-name. That field name then receives one of the field values that belong to that parameter.

To extend the existing context to a Transformed context, a **TRANSFORM** line is being added, preceding the SQL SELECT query.

Also see the Relations - Transformed columns and Templates - Transform articles

Coded relation functions

Some relations are using a subroutine, which is written in code to perform more complicated actions. These are usually created for specific cases per customer request. The format is like this:

```
name_of_subroutine(<variable(s)>)
```

These subroutines can be added as a plug-in into `/opt/yce/bin/context_functions.pl`. Examples are `node_interfaces(<hostname>)` and `vlan_member_list(<hostname>)`.

These subroutine calls replace the “Relation_query” in the relation definition.

The custom reports cannot use the subroutines defined in context_functions.pl.