

Extending NOC with Custom Fields

Upcoming NOC 0.7(4) contains new clue functionality - *Custom Fields*.

You can add custom fields to virtually any NOC data table to store your custom data and to organize workflow.

Lets consider a simple example - we need to store customer id attached to VLAN.

It is recommended to stop all NOC processes except noc-web before creating custom field

In *Main > Setup > Custom Fields* press "Add" and fill form

The screenshot shows the NOC web interface. The left sidebar contains a navigation menu with categories like Main, Reports, and Setup. Under Setup, 'Custom Fields' is selected. The main content area displays the 'Change Custom Fields' form. The form includes fields for 'Table' (set to 'vc_vc'), 'Name' (set to 'customer'), 'Label' (set to 'Customer'), 'Type' (set to 'String'), 'Max. Length' (set to '64'), and 'Regexp'. There are also checkboxes for 'Is Active', 'Is Indexed', 'Is Searchable', 'Is Filtered', and 'Is Hidden'. The 'Is Active' checkbox is checked. The footer of the interface shows the copyright notice: ©2007-2012, nocproject.org.

Press "Save".

New table field will be created:

```

/opt/noc$ psql noc
noc=# \d vc_vc

```

```

Table "public.vc_vc"
-----
Column          | Type          | Modifiers
-----
id              | integer      | not null default nextval('vc_vc_id_seq'::regclass)
vc_domain_id   | integer      | not null
l1              | integer      | not null
l2              | integer      | not null default 0
description     | character varying(256)
name           | character varying(64) | not null
tags           | character varying(255)
style_id       | integer
project        | character varying(256)
cust_customer  | character varying(64)

Indexes:
"vc_vc_pkey" PRIMARY KEY, btree (id)
"vc_vc_l1" UNIQUE, btree (vc_domain_id, l1, l2)
"vc_vc_vc_domain_id_21725b7fb0adaf81_uniq" UNIQUE CONSTRAINT, btree (vc_domain_id, name)
"vc_vc_customer" btree (cust_customer)
"vc_vc_project" btree (project)
"vc_vc_project_like" btree (project varchar_pattern_ops)
"vc_vc_style_id" btree (style_id)
"vc_vc_vc_domain_id" btree (vc_domain_id)

Foreign-key constraints:
"style_id_refs_id_34edbd6f1eccde9a" FOREIGN KEY (style_id) REFERENCES main_style(id) DEFERRABLE INITIALLY DEFERRED
"vc_domain_id_refs_id_2ad42705d2e81e25" FOREIGN KEY (vc_domain_id) REFERENCES vc_vcdomain(id) DEFERRABLE INITIALLY DEFERRED

```

Note the *cust_customer* field has been created in *vc_vc* table, so the Custom Field is first-class citizen in PostgreSQL tables and can be referred as *cust_<name>* in any SQL statement.

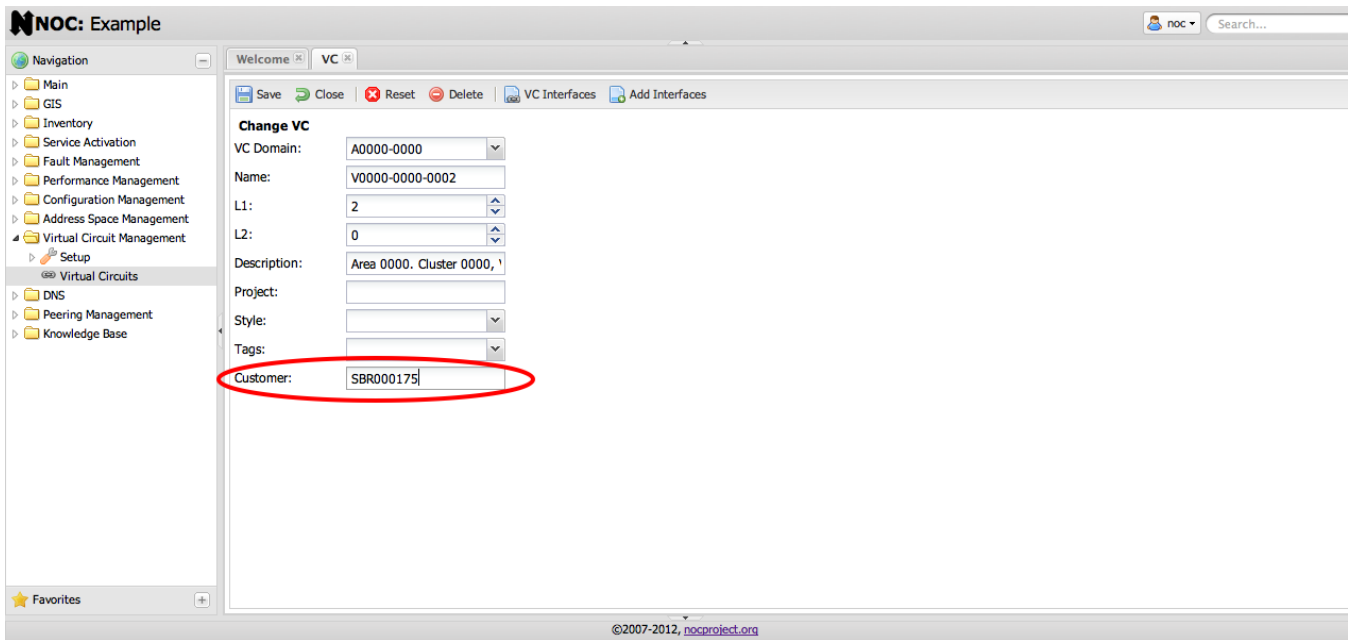
Next, restart NOC processes to apply changes.

Lets check our *Virtual Circuits* form:

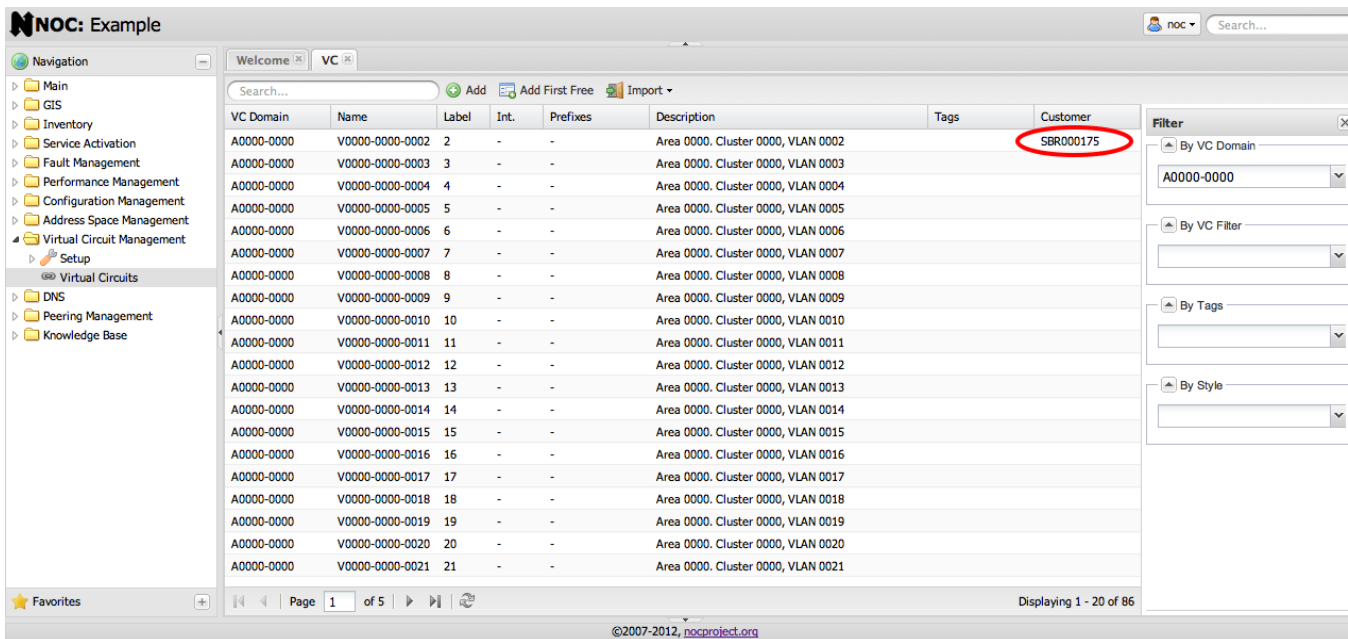
The screenshot shows the NOC web interface with a table of Virtual Circuits. The table has columns: VC Domain, Name, Label, Int., Prefixes, Description, Tags, and Customer. A red arrow points to the 'Customer' column header. On the right, a 'Columns' filter menu is open, showing a list of columns with checkboxes. The 'Customer' checkbox is checked, and a red arrow points to it.

VC Domain	Name	Label	Int.	Prefixes	Description	Tags	Customer
A0000-0000	V0000-0000-0002	2	-	-	Area 0000. Cluster 0000, VLAN 0002		
A0000-0000	V0000-0000-0003	3	-	-	Area 0000. Cluster 0000, VLAN 0003		
A0000-0000	V0000-0000-0004	4	-	-	Area 0000. Cluster 0000, VLAN 0004		
A0000-0000	V0000-0000-0005	5	-	-	Area 0000. Cluster 0000, VLAN 0005		
A0000-0000	V0000-0000-0006	6	-	-	Area 0000. Cluster 0000, VLAN 0006		
A0000-0000	V0000-0000-0007	7	-	-	Area 0000. Cluster 0000, VLAN 0007		
A0000-0000	V0000-0000-0008	8	-	-	Area 0000. Cluster 0000, VLAN 0008		
A0000-0000	V0000-0000-0009	9	-	-	Area 0000. Cluster 0000, VLAN 0009		
A0000-0000	V0000-0000-0010	10	-	-	Area 0000. Cluster 0000, VLAN 0010		
A0000-0000	V0000-0000-0011	11	-	-	Area 0000. Cluster 0000, VLAN 0011		
A0000-0000	V0000-0000-0012	12	-	-	Area 0000. Cluster 0000, VLAN 0012		
A0000-0000	V0000-0000-0013	13	-	-	Area 0000. Cluster 0000, VLAN 0013		
A0000-0000	V0000-0000-0014	14	-	-	Area 0000. Cluster 0000, VLAN 0014		
A0000-0000	V0000-0000-0015	15	-	-	Area 0000. Cluster 0000, VLAN 0015		
A0000-0000	V0000-0000-0016	16	-	-	Area 0000. Cluster 0000, VLAN 0016		
A0000-0000	V0000-0000-0017	17	-	-	Area 0000. Cluster 0000, VLAN 0017		
A0000-0000	V0000-0000-0018	18	-	-	Area 0000. Cluster 0000, VLAN 0018		
A0000-0000	V0000-0000-0019	19	-	-	Area 0000. Cluster 0000, VLAN 0019		
A0000-0000	V0000-0000-0020	20	-	-	Area 0000. Cluster 0000, VLAN 0020		
A0000-0000	V0000-0000-0021	21	-	-	Area 0000. Cluster 0000, VLAN 0021		

Click on arrow at the column header, and in Columns menu check *Customer*. You will see new *Customer* column in grid. Lets change customer for VLAN.



Note the *Customer* field is first-class citizen in the form. Press "Save" button



We can see the customer ID in the grid. Lets check on database level:

```

/opt/noc$ psql noc
noc=# SELECT * FROM vc_vc WHERE cust_customer = 'SBR000175';
 id | vc_domain_id | l1 | l2 | description | name | tags | style_id | project
 | cust_customer
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
 22 |          4 | 2 | 0 | Area 0000. Cluster 0000, VLAN 0002 | V0000-0000-0002 | | | |
 | SBR000175
(1 row)

```

Next, lets check JSON interface

```
/opt/noc$ http -a noc:noc GET 'http://127.0.0.1:8000/vc/vc/?customer=SBR000175'  
HTTP/1.1 200 OK  
Content-Language: en-us  
Expires: 0  
Vary: Accept-Language, Cookie  
Pragme: no-cache  
Cache-Control: no-cache  
Content-Type: text/json; charset=utf-8  
Content-Length: 306  
Server: TornadoServer/2.2  
[  
  {  
    "customer": "SBR000175",  
    "description": "Area 0000. Cluster 0000, VLAN 0002",  
    "id": 22,  
    "interfaces_count": "-",  
    "l1": 2,  
    "l2": 0,  
    "name": "V0000-0000-0002",  
    "prefixes": "-",  
    "project": "",  
    "row_class": "",  
    "style": null,  
    "style__label": "",  
    "tags": [],  
    "vc_domain": 4,  
    "vc_domain__label": "A0000-0000"  
  }  
]
```

So the custom fields are first-class citizens in JSON interface too opening clean way for third-party systems integration.

Finally, lets check ORM level

```
/opt/noc$ ./noc shell  
Python 2.7.3 (default, May 5 2012, 17:19:28)  
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin  
Type "help", "copyright", "credits" or "license" for more information.  
(InteractiveConsole)  
>>> from noc.vc.models import VC  
>>> vc = VC.objects.get(customer="SBR000175")  
>>> vc  
<VC: A0000-0000 2: V0000-0000-0002>  
>>> vc.customer  
u'SBR000175'
```

So the custom fields can be used in pyRules, scripts and triggers just like built-in fields.