Tech Tip 5 - Auto Generate Service Groups

Tech Tip 5 (11/2016) - Using the sg_autocreate script to automatically create service groups

The use case for this script is as follows:

- You need to generate service groups in the Foundation database so that they show up in the UI.
- You have a lot of hosts and services, so using the UI to create these servicegroups is time consuming and tedious.
- You have changes to the hosts-service combinations that you want represented in the service groups, and these happen regularly enough that it is desirable to automate the service group creation.

This script will make it possible to create service groups automatically on the commit of the configuration, using data from the existing hosts and services.
While you can run this script at the command line and so integrate it into your existing provisioning systems, we have included an example of calling it from the commit process using the MonarchCallOut.pm integration module.

The script itself is heavily modular and commented, making it an ideal example for extending to other, similar functions as needed. We welcome your contributions in this regard, as always, and your suggestions for improvement and more use cases.

Here's how to install it

Download all the files:

Name	Size	Creator	Creation Date	Comment
sg_autocreate	68 kB	Thomas Stocking	Nov 07, 2016 14:56	
sg_autocreate.conf	5 kB	Thomas Stocking	Nov 07, 2016 14:55	
sg_autocreate_servicegroups.conf	2 kB	Thomas Stocking	Nov 07, 2016 14:56	
agent_id	0.1 kB	Thomas Stocking	Nov 07, 2016 14:56	
MonarchCallOut.pm	1 kB	Thomas Stocking	Nov 07, 2016 14:56	

Place the files on the GroundWork server. Put:
sg_autocreate
sg_autocreate.conf
sg_autocreate_servicegroups.conf

in a directory on the GroundWork server where they can be accessed by the nagios user, e.g. **/usr/local/groundwork/scripts**. This directory is where it is set up to go, so if you use something else, you will need to change the paths in the config files.
You can put **agent_id** in there too, if you like. You only need to run it once, usually, as part of setup.

Make all the files owned by user nagios:

```
chown nagios.nagios /usr/local/groundwork/scripts/sg_autocreate*
```

Make sure that the **sg_autocreate** script is executable:

```
chmod +x /usr/local/groundwork/scripts/sg_autocreate
```

Do the same for the **agent_id** script.

Use the **agent_id** script to generate a unique id for the creation (and deletion) of the servicegroups you make using this script. You can use any string, as long as it is unique, but this makes a nice example.

```
./agent_id
agent_id = "59e53d98-a526-11e6-a10c-8170f6c82928"
```

The generated string is randomized enough to be used as a unique agent identifier in the foundation database. Edit the file **/usr/local/groundwork/scripts/sg_autocreate.conf** and change the default agent ID to the one you generated, like this:

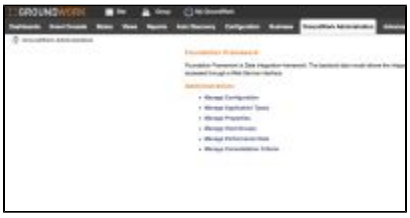
```
vi /usr/local/groundwork/scripts/sg_autocreate.conf
...
agent_id = "59e53d98-a526-11e6-a10c-8170f6c82928"
```

Also, while editing this file optionally change the application type to one you will recognize later. You can also just use the default:

```
application_type = "AUTOGROUP"
```

Save the changes to the file.

Now you will need to add the application type to foundation. Let's say you used "AUTOGROUP" like the example above, though you can call it anything you like. Log in to the web GUI as an administrative user, and click on GroundWork Administration, choosing the first item in the menu, Foundation. That will show you this screen:



Click on Manage Application Types, then click the grey Add Application Type button, and enter the app type you set in the file, e.g. AUTOGROUP. You should see it listed in the resulting screen:

Choose your service group members

Next, decide what the membership of the service group will be. You can have multiple services in one group or just one. If you are making use of service instances, you can choose to use the base name of the service or the suffix, depending on what you want. If you use the base name, all instances will be included. If you use the suffix, all services with that suffix will be included.

You make these choices by editing the file /usr/local/groundwork/sg_autocreate_servicegroups.conf. There are examples in the file for you to work from.
For instance, if you wanted a service group called "App_checks", which contained all instances of the service "app_check" on all hosts, then you would put in:

```
<service app_check>
     service_group = "App_checks"
     service_group_description = "App checks on all servers"
 </service>
```

If you have a service like interface_status with instances named like _01, _02, _03, etc, you could capture them all with the configuration:

```
<service interface_status>
     service_group = "interface_checks"
     service_group_description = "interface checks on all systems"
 </service>
```

If, however, you have services with instances named _myapp, and the base name of the service might differ, you can create a servicegroup with all the instances named _myapp with:

```
<service_instance _myapp>
    service_group = "Service_checks_for_MyApp"
    service_group_description = "All the MyApp checks"
</service_instance>
```

the restrictions on service group names apply. No spaces are allowed.

Run the script and create the groups

Once you have it looking the way you want, you can run the script as user nagios:

```
su - nagios
/usr/local/groundwork/scripts/sg_autocreate -m

(date stamp) Service group auto-creation script (version 1.0.0) is starting up.
INFO:  Running with options:  sg_autocreate -m
NOTICE:  Processing 1 <service> element and 1 <service_instance> element representing servicegroups.
NOTICE:  Processed 2 configured servicegroups.
```

Then check the Status view for your new service groups. They will be there within a few seconds.

Cleaning up and formalizing

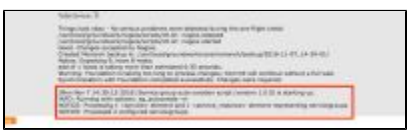
If you want to **remove** any of the groups you create in this way, just comment them out of the /usr/local/groundwork/sg_autocreate_servicegroups.conf file. They will be removed the next time you run the script.

The agent ID and application type are associated with these groups in the database. If you overwrite these values in the config file, removing the groups with the script will no longer work, and you would need to remove them by modifying the database directly.

What if you want to run the script each time you commit changes to the configuration? That's where MonarchCallOut.pm comes in. Just copy the supplied version over /usr/local/groundwork/core/monarch/lib/MonarchCallOut.pm. Make sure it is still owned by user nagios when done:

```
#cp MonarchCallOut.pm /usr/local/groundwork/core/monarch/lib/MonarchCallOut.pm
#chown nagios.nagios /usr/local/groundwork/core/monarch/lib/MonarchCallOut.pm
```

Now whenever you run a commit, you will see the output from the script at the end. This will add some processing time. Depending on the size of your deployment, you may or may not want to do it this way.



See also:

The APIs we are using are all documented here:

- dassMonarch
- Foundation REST API

We hope you find this useful and informative. Thanks for reading!